4.5 Order Manager Subsystem Overview

The Order Manager subsystem (OMS) manages all orders arriving via the Data Management subsystem’s V0 Gateway (V0 GTWAY), (i.e., submitted by EDG and ECHO users), as well as data orders submitted by the spatial subscription server (NSBRV). There are plans to interface all ECS components that handle user orders with the new Order Management Service in the future, though not during Synergy III.

The capability includes a new server (the Order Manager Server) to which the data distribution orders are submitted and which then distribute the orders to the appropriate ECS services (i.e., PDS or SDSRV), depending on whether the request is for media or electronic distribution. It also includes a new database that stores all order information persistently as soon as an order is received by ECS and before its receipt is acknowledged. This allows operators to resubmit an order if it encounters errors downstream, and allows the Order Management Service to perform some up front checks on the order and alert the operators if their intervention is needed.

Order Manager Subsystem Context

Figure 4.5-1 is the Order Manager Subsystem context diagram. The diagram shows the events sent to the Order Manager Subsystem and the events the Order Manager Subsystem sends to other subsystems. Table 4.5-1 provides descriptions of the interface events shown in the Order Manager Subsystem context diagram.

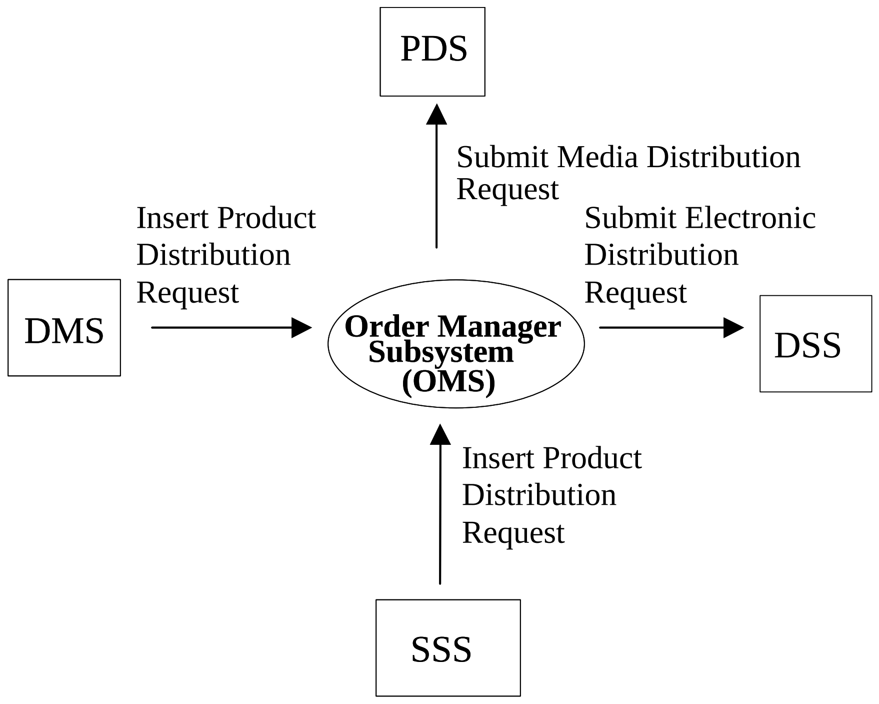

PDS
Submit Media Distribution Request
Insert Product Distribution Request
Submit Electronic Distribution Request
DMS
Order Manager Subsystem (OMS)
DSS
Insert Product Distribution Request
SSS

Figure 4.5-1. Order Manager Subsystem Context Diagram

Table 4.5-1. Order Manager Subsystem Interface Events

Event	Interface Event Description
Submit Media Distribution Request	The Order Manager Subsystem (OMS) submits all product distribution requests that specify physical media types to the **PDS** using the V0 ODL protocol.
Submit Electronic Distribution Request	The OMS submits all product distribution requests, which specify electronic media types (e.g., FTPPUSH, FTPPULL) to the **Data Server Subsystem (DSS**).
Insert Product Distribution Request	The **Data Management Subsystem (DMS)** and **Spatial Subscription Server (SSS)** insert product distribution requests in the Order Manager Data Base Management System through the OMS client library.

Order Manager Subsystem Structure

- The ORDER MANAGER Subsystem consists of two CSCIs, OMSRV and OMGUI. The Order Manager Server (OMSRV) is a software configuration item. The Order Manager Server receives product distribution requests and submits them to the appropriate ECS component based upon the media type specified for the request. ORDER MANAGER Subsystem information is stored persistently in a relational Database Management System (DBMS). The Order Manager GUI (OMGUI) is used to monitor and control the operations of the Order Manager Server. In addition, the OMGUI is used to respond to Operator Intervention Requests generated by the Order Manager Server.

Use of COTS in the Order Manager Subsystem

- RogueWave’s Tools.h++

  The Tools.h++ class libraries are used by the OMS to provide basic functions and objects such as strings and collections. These libraries must be installed with the OMS software for any of the OMS processes to run.

- RogueWave’s DBTools.h++

  The DBTools.h++ C++ class libraries are used to interact with the Sybase database SQL server. The use of DBTools buffers the OMS processes from the relational database used. These libraries must be installed with the OMS software for the Order Manager Server to run and allow client processes to perform queries of Order Manager database information.

- Sybase Open Client / CT_LIB

  The Sybase Open Client provides access between OMS custom code and the Sybase SQL Server DBMS.

- Sendmail

  A product used to send electronic mail between users consisting of mailboxes, user agents, transfer agents and delivery agents. A mailbox is a file, or possibly a directory of files, where incoming messages are stored. A mail user agent, or MUA, is an application run directly by a user. Mail transfer agents (MTAs) are used to transfer messages between machines. Delivery agents are used to place a message into a user's mailbox.

- Sybase Server

  The Sybase SQL server provides access for OMS to insert, update and delete Product Distribution Requests, OMS configurations, and Operator Interventions. The Sybase SQL Server must be running during operations for the OMS to process Product Distribution Requests.

4.5.1 Order Manager Subsystem Software Description

4.5.1.1 Order Manager Server CSCI Functional Overview

The Order Manager Server (OMSRV) CSCI is one process, the Order Manager Server, which interacts with the Order Manager Database, the Product Distribution System (PDS), and the Science Data Server (SDSRV). The V0 Gateway and the Spatial Subscription Server (SSS) submit Product Distribution Requests to the OMS. These requests are validated and if valid are submitted to the PDS or SDSRV. For invalid requests, an Operator Intervention is generated. DAAC OPS personnel can use the Order Manager GUI to correct and resubmit the request. In response to an intervention, the Operator can also generate an email message, which is sent to the user by the Order Manager Server.

4.5.1.2 Order Manager Server CSCI Context

Figure 4.5-2 is the Order Manager Server CSCI context diagrams. The diagrams show the events sent to the Order Manager Server CSCI and the events the Order Manager Server CSCI sends to other CSCIs. Table 4.5-2 provides descriptions of the interface events shown in the Order Manager Server CSCI context diagrams.

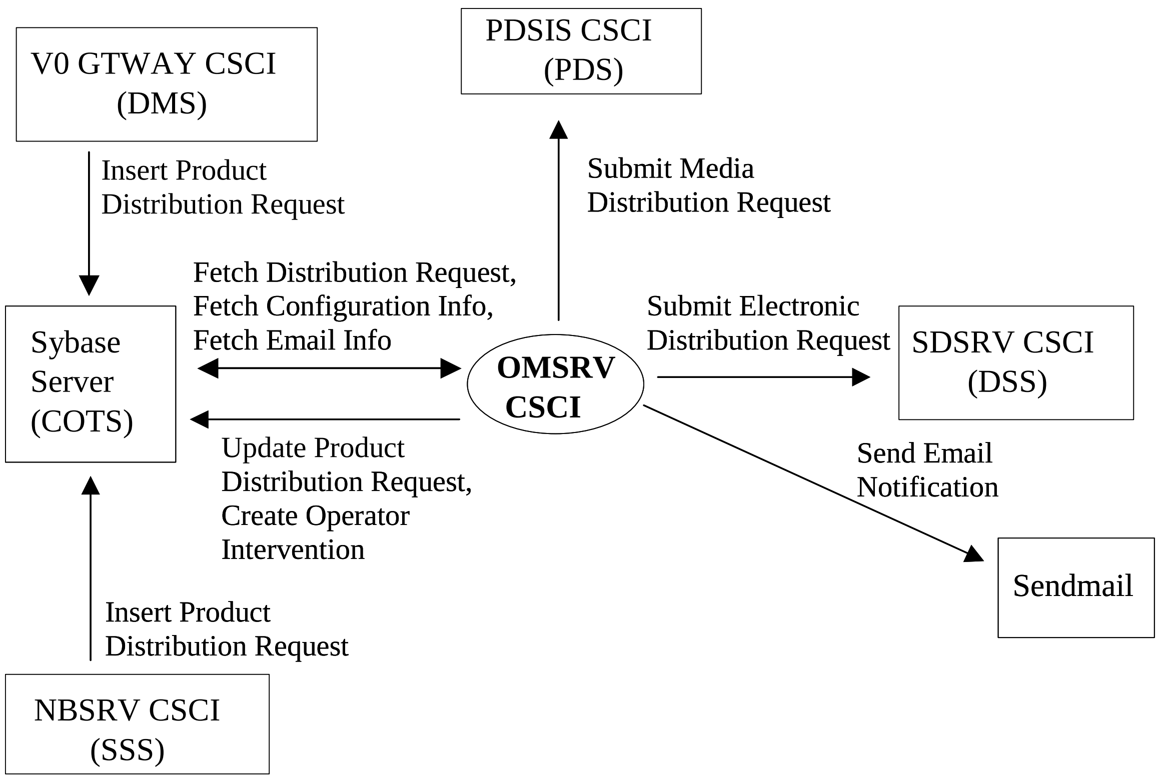

V0 GTWAY CSCI (DMS)
PDSIS CSCI (PDS)
Insert Product Distribution Request
Submit Media Distribution Request
Fetch Distribution Request, Fetch Configuration Info, Fetch Email Info
Submit Electronic Distribution Request
Sybase Server (COTS)
OMSRV CSCI
SDSRV CSCI (DSS)
Update Product Distribution Request, Create Operator Intervention
Send Email Notification
Sendmail
Insert Product Distribution Request
NBSRV CSCI (SSS)

Figure 4.5-2. Order Manager Server CSCI Context Diagram

Table 4.5-2. Order Manager Server CSCI Interface Events (1 of 2)

Event	Interface Event Description
Submit Media Distribution Request	The OMSRV CSCI submits requests for product distribution requests on physical media to the **PDSIS CSCI**.
Submit Electronic Distribution Request	The OMSRV CSCI submits electronic product distribution requests to the **Science Data Server (SDSRV) CSCI**.
Send Email Notification	The OMSRV CSCI sends email notifications to via the **Sendmail** COTS package.
Insert Product Distribution Request	The **V0 Gateway (V0 GTWAY) CSCI** and **Spatial Subscription Server (NBSRV) CSCI** inserts product distribution requests into the Sybase Server (Order Manager database).
Update Product Distribution Request	The OMSRV updates the Product Distribution Request information in the **Sybase Server (Order Manager database)**.
Create Operator Intervention	The OMSRV creates new Operator Intervention requests in the **Sybase Server (Order Manager database)**.
Fetch Distribution Requests	Retrieves information associated with a Product Distribution Request from the **Sybase Server (Order Manager database)**.

Table 4.5-2. Order Manager Server CSCI Interface Events (2 of 2)

Event	Interface Event Description
Fetch Configuration Info	Retrieves the OMSRV Configuration information from the **Sybase Server (Order Manager database)**.
Fetch Email Info	Retrieves information related to an operator intervention that is required to generate an email notification from the **Sybase Server (Order Manager database)**.

4.5.1.3 Order Manager Server CSCI Architecture

Figure 4.5-3 is the Order Manager Server (OMSRV) CSCI architecture diagram. The diagram shows the events sent to the OMSRV CSCI processes and the events the OMSRV CSCI processes send to other processes.

The OM Server CSCI consists of one process. This process is the EcOmOrderManager process.

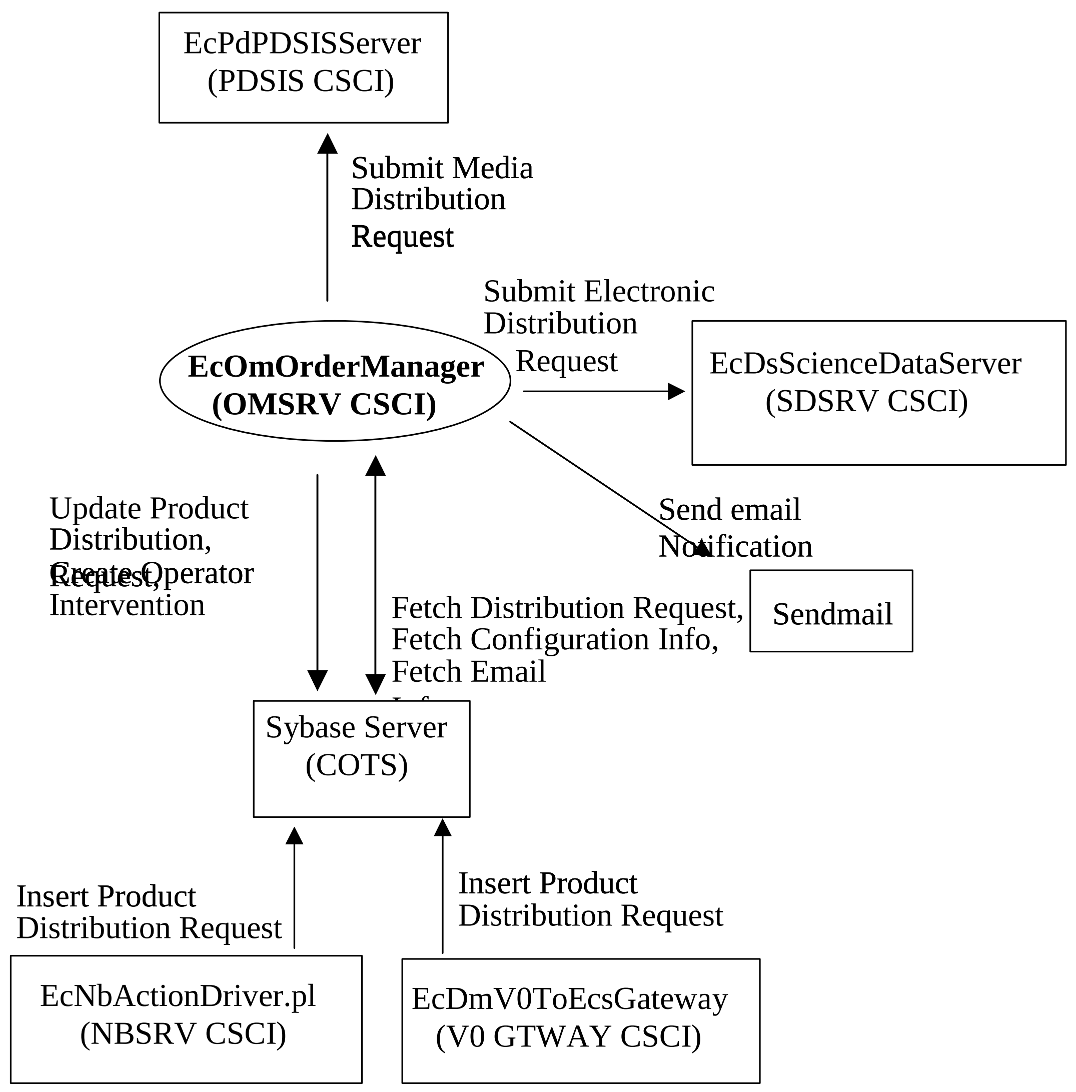

EcPdPDSISServer (PDSIS CSCI)
Submit Media Distribution Request
Submit Electronic Distribution Request
EcOmOrderManager (OMSRV CSCI)
EcDsScienceDataServer (SDSRV CSCI)
Update Product Distribution, Request, Create Operator Intervention
Send email Notification
Fetch Distribution Request, Fetch Configuration Info, Fetch Email Info
Sendmail
Sybase Server (COTS)
Insert Product Distribution Request
Insert Product Distribution Request
EcNbActionDriver.pl (NBSRV CSCI)
EcDmV0ToEcsGateway (V0 GTWAY CSCI)

Figure 4.5-3. Order Manager Server CSCI Architecture Diagram

4.5.1.4 Order Manager Server CSCI Process Description

Table 4.5-4 provides descriptions of the processes shown in the OMSRV CSCI architecture diagram.

Table 4.5-4. OMSRV CSCI Process

Process	Type	COTS/ Developed	Functionality
EcOmOrderManager	Server	Developed	The Order Manager Server retrieves product distribution requests from the Order Manager database and dispatches them to the PDS or the SDSRV according to the media type of each request.

4.5.1.5 Order Manager Server CSCS Interface Description

Table 4.5-5 provides descriptions of the interface events shown in the Order Manager Server (OMSRV) CSCI architecture diagram.

Table 4.5-5. Order Manager Server CSCI Process Interface Events (1 of 2)

Event	Event Frequency	Interface	Initiated By	Event Description
Submit Media Distribution Request	One or more granules per service request	*Process:* EcPdPDSISServer *Library:* Pdsis.jar *Class:* ECSPDSServer	*Process:* EcOmOrderManager *Library:* V0XV0Staticlibs *Class:* OmPDSIf	The EcOmOrderManager submits requests for product distribution requests on physical media to the EcPdPDSISServer.
Submit Electronic Distribution Request	One or more granules per service request	*Process:* EcDsScienceDataServer *Library:* ? *Class:* ?	*Process:* EcOmOrderManager *Executable:* EcOmSrCLI *Library:* DsClientSideLibs *Class:* OmSdsrvIf	The EcOmOrderManager submits requests for Electronic product distribution requests to the EcDsScienceDataServer.
Send email notification	One per product request	*Process:* sendmail (COTS)	*Process:* EcOmOrderManager *Class:* OmSrEmailRequest	The EcOmOrderManager sends email notifications to end-users.
Fetch Distribution Request	One request per configurable interval	*Process:* Sybase Server (COTS) Data base table: OmRequest	*Process:* EcOmOrderManager *Library:* Sybase Ct-library *Classes:* OmSrDbInterface, OmSrDistributionRequest	The EcOmOrderManager retrieves information associated with a product distribution request from the database.

Table 4.5-5. Order Manager Server CSCI Process Interface Events (2 of 2)

Event	Event Frequency	Interface	Initiated By	Event Description
Fetch configuration info	Once per startup/ Once per configurable interval	*Process:* Sybase Server (COTS) Data base table: OmConfigParameter	*Process:* EcOmOrderManager *Library:* Sybase Ct-library *Class:* OmSrDbInterface	The EcOmOrderManager retrieves configuration information from the database.
Fetch email info	Once per configurable interval	*Process:* Sybase Server (COTS) Data base table: OmActionQueue	*Process:* EcOmOrderManager *Library:* Sybase Ct-library *Classes:* OmSrDbInterface, OmSrDistributionRequest, OmSrEmailRequest	The EcOmOrderManager retrieves information related to an operator intervention required to generate an email notification.
Insert Product Distribution Request	One insertion per service request	*Process:* Sybase Server (COTS) Data base table: OmRequest	*Process:* EcDmV0ToEcsGateway *Library:* OmClientlib *Classes:* OmSrDbInterface, OmSrV0InputIf	The EcDmV0ToEcsGateway inserts product distribution requests into the Order Manager database.
Update Product Distribution Request	One update per request	*Process:* Sybase Server (COTS) Data base table: OmRequest	*Process:* EcOmOrderManager *Library:* Sybase Ct-library *Classes:* OmSrDbInterface, OmSrDistributionRequest	The EcOmOrderManager updates existing product distribution requests in the Order Manager database.
Create Operator Intervention	One per operator request	*Process:* Sybase Server (COTS) Data base table: OmActionQueue	*Process:* EcOmOrderManager *Library:* Sybase Ct-library *Class:* OmSrDbInterface	The EcOmOrderManager creates a new Operator Intervention request in the database.

4.5.1.6 Data Stores

Table 4.5-6 provides descriptions of the data stores used by the Spatial Subscription Server (EcNbActionDriver.pl) and V0 Gateway (EcDmV0ToEcsGateway) in the Order Manager Server CSCI architecture diagram. V0 Gateway uses only the OmRequest, OmRequestOptions and OmGranule data stores. The Spatial Subscription Server uses all of the data stores in this table.

Table 4.5-6. Order Manager Server (OMSRV) CSCI Data Stores

Data Store	Type	Functionality
OmRequest	Oracle	This data store holds each distribution request received by an OMS client.
OmRequestOptions	Oracle	This data store stores the optional distribution options per request.
OmGranule	Oracle	This data store holds the granule specific contained within each distribution request.
OmStatus	Oracle	This data store holds the static values for the status of requests, granules, notifications, actions and bundled orders.
OmBundlingOrder	Oracle	This data store holds the information pertaining to bundled orders for which subscriptions are submitted against.
OmConfigParameter	Oracle	This data store holds the dynamic configuration parameters for the OmServer. These parameters can be modified without rebooting the OmServer.
OmActionQueue	Oracle	This data store holds the current set of actions queued.
OmMediaType	Oracle	This data store holds those parameters specific for each media type.

4.5.1.7 Order Manager Hardware

See the CSS HW CI section 2 for the hardware description.

4.6 Planning Subsystem Overview

The Planning Subsystem (PLS) manages the data production activities at ECS sites in support of the operations staff by providing the following capabilities:

- Identifies the data processing tasks (via data processing requests) performed by a site.
- Generates the data production plans for scheduling the identified processing tasks according to different production rules, which define how a particular Product Generation Executive (PGE) is to be run.
- Coordinates data production with the DSS and the DPS to achieve an automated production system.

Planning Subsystem Context Diagram

Figure 4.6-1 is the context diagrams for the PLS. The diagrams show the events sent to other SDPS and CSMS subsystems and the events the PLS receives from other SDPS and CSMS subsystems. Table 4.6-1 provides descriptions of the interface events shown in the Planning Subsystem Context Diagrams.

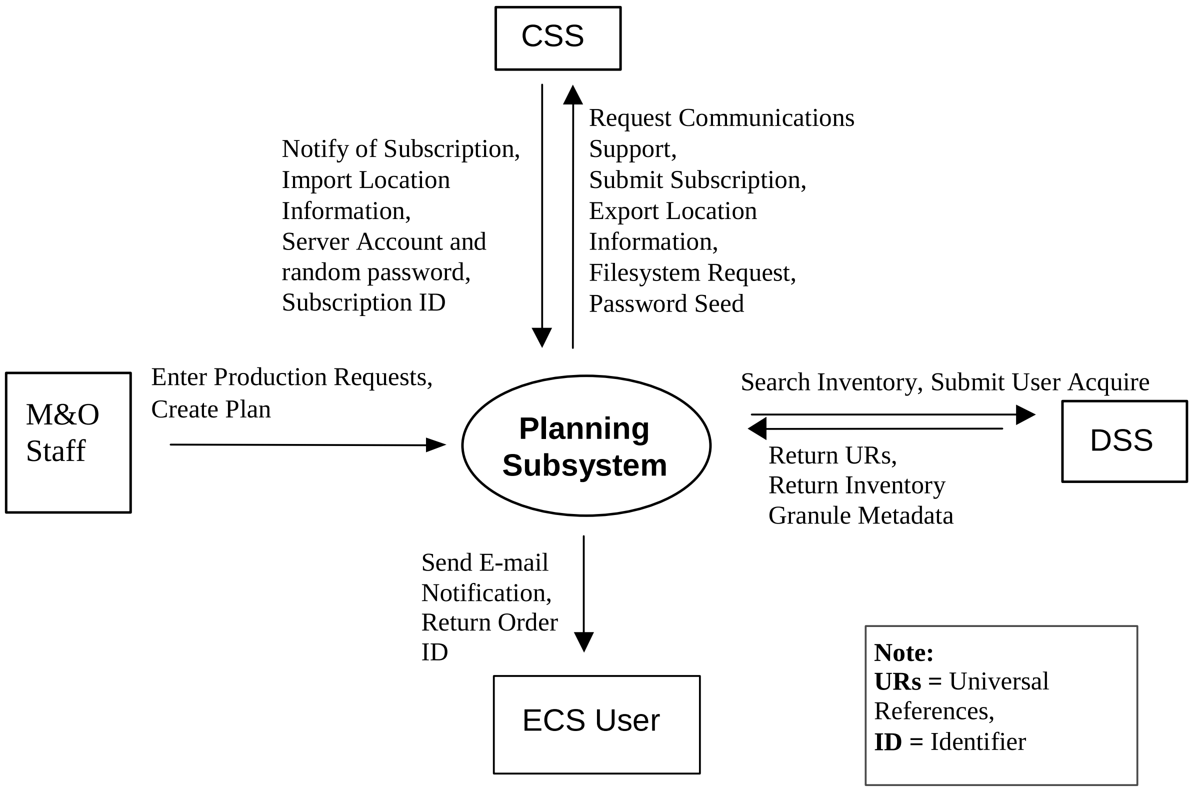
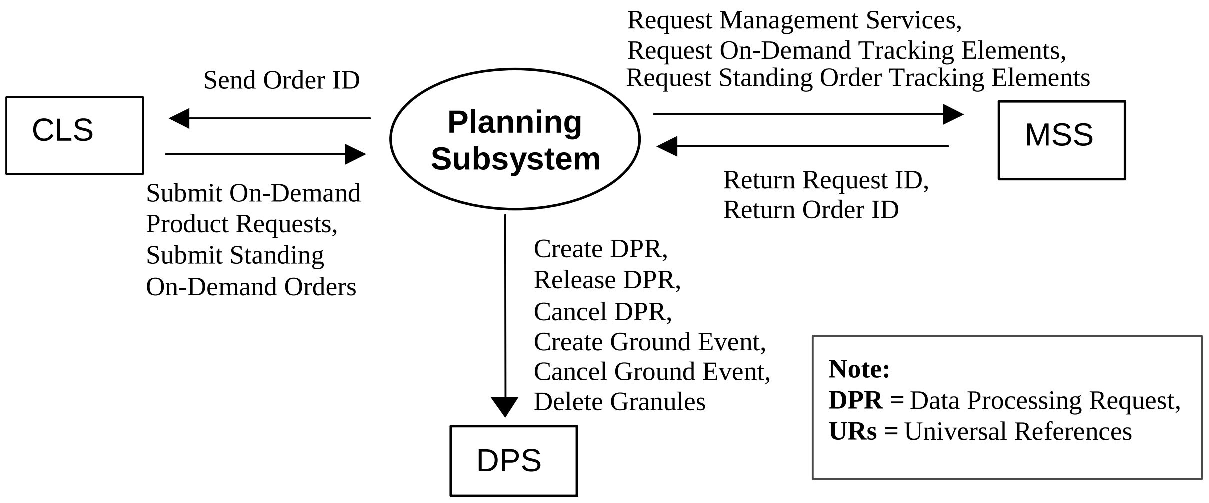

CSS
Notify of Subscription, Import Location Information, Server Account and random password, Subscription ID
Request Communications Support, Submit Subscription, Export Location Information, Filesystem Request, Password Seed
M&O Staff
Enter Production Requests, Create Plan
Planning Subsystem
Search Inventory, Submit User Acquire
DSS
Return URs, Return Inventory Granule Metadata
Send E-mail Notification, Return Order ID
ECS User
Note: URs = Universal References, ID = Identifier

Figure 4.6-1. Planning Subsystem Context Diagram



Request Management Services,
Request On-Demand Tracking Elements,
Request Standing Order Tracking Elements
Send Order ID
CLS
Planning Subsystem
MSS
Return Request ID,
Return Order ID
Submit On-Demand Product Requests,
Submit Standing On-Demand Orders
Create DPR,
Release DPR,
Cancel DPR,
Create Ground Event,
Cancel Ground Event,
Delete Granules
DPS
Note:
DPR = Data Processing Request,
URs = Universal References

Figure 4.6-1. Planning Subsystem Context Diagram (cont.)

Table 4.6-1. Planning Subsystem Interface Events (1 of 3)

Event	Interface Event Description
Request Communications Support	The **CSS** provides a library of services available to each SDPS and CSMS subsystem. The subsystem services required to perform specific assignments are requested from the CSS. These services include: • CCS Middleware Support • Database Connection Services • Network & Distributed File Services • Name/Address Services • Password Services • Server Request Framework (SRF) • Universal Reference (UR) • Error/Event Logging • Message Passing • Fault Handling Services • Mode Information • Query Registry - Retrieving the requested configuration attribute-value pairs from the Configuration Registry
Submit Subscription	The PLS creates a subscription, sent to the **CSS**, using the advertisement for subscribing to an insert event for an ESDT. In response, PLS receives a corresponding subscription identifier.

Table 4.6-1. Planning Subsystem Interface Events (2 of 3)

Event	Interface Event Description
Export Location Information	The PLS stores physical and logical server location information in the **CSS**.
Filesystem Request	The PLS requests ECS files and directories via an established mount point from the **CSS**. The CSS makes the storage device(s) and its data accessible for use by clients.
Password Seed	The PLS requests an account and provides a password to the **CSS**.
Search Inventory	The PLS sends inventory search or inspect requests to the **DSS** to search the SDPS inventory/archives (granules). In response, the PLS receives URs for the respective granules satisfying the search.
Submit User Acquire	The PLS submits an acquire command to the **DSS** on behalf of the user. The user gets a response via the DSS upon data distribution.
Return URs	The PLS receives Earth Science Data Type (ESDT) Universal References (URs) for the granules from the **DSS**.
Return Inventory Granule Metadata	The PLS receive the inventory granule metadata identifying the scene within the granule based on an inventory search request sent to the **DSS**.
Send E-mail Notification	The PLS sends e-mail notification to the **user** when a standing order granule arrives and when a standing order expires.
Return Order ID	The PLS sends an order id to the **user** to track his on-demand product or order requests.
Enter Production Requests	The **M&O staff** request production by selecting a PGE type and the time duration for the PGE to process the input data in the PLS.
Create Plan	The **M&O staff** issue commands to control the creation and activation of a data production plan to the PLS.
Notify of Subscription	A message passing callback in the PLS subscription manager is called, by the **CSS**, with the UR of the granule inserted into the Data Server as one of the calling parameters.
Import Location Information	The PLS requests server location information from the **CSS**.
Server Account and random password	The PLS receives an account and random password from the **CSS** after providing a password seed to establish an account.
Subscription ID	The PLS receives a subscription identifier from the **CSS** after submitting a subscription.
Request Management Services	**System startup and shutdown -** Please refer to the release-related, current version of the Mission Operations Procedures for the ECS Project document (611) and the current ECS Project Training Material document (625), identified in Section 2.2.1 of this document.
Request On-Demand Tracking Elements	The PLS sends product requests to the **MSS** for on-demand orders.
Request Standing Order Tracking Elements	The PLS sends requests to the **MSS** for standing orders of ASTER high level on-demand products.

Table 4.6-1. Planning Subsystem Interface Events (3 of 3)

Event	Interface Event Description
Return Request ID	The PLS receives the request identifier from the **MSS** for on-demand tracking elements requests.
Return Order ID	The PLS receives the order identifier from the **MSS** for standing order tracking elements requests.
Create DPR	The PLS sends, to the **DPS**, the Data Processing Request Identification (dprId) and whether the DPR is waiting for external input data.
Release DPR	The PLS sends the dprId to the **DPS** for DPR release.
Cancel DPR	The PLS sends a request to cancel the dprId to the **DPS** for the deletion of a DPR.
Create Ground Event	The PLS sends the ground event id, resource id, and start time to the **DPS** to create a ground event to perform maintenance activities on data processing resources.
Cancel Ground Event	The PLS sends the ground event id, resource id, and start time to the **DPS** to cancel a ground event.
Delete Granules	The PLS sends requests to the **DPS** to delete granules associated with cancelled DPRs.
Submit On-Demand Product Requests	The PLS receives on-demand requests from the **CLS**. As a result, the user receives an Order ID. The user receives a notification when the request is processed.
Submit Standing On-Demand Orders	The PLS receives standing on-demand orders from the **CLS**.
Send Order ID	The PLS returns the order identifier to the **CLS**.

The following paragraphs describe the relationships between the PLS and other SDPS subsystems.

DPS Interface

The PLS uses a database link with the DPS Processing CSCI to describe the Product Generation Executives (PGEs) needed to fulfill the production goals. A Data Processing Request (DPR) describes a PGE run to the DPS. A DPR describes the specific input granules, output filenames, and run-time parameters for a PGE, as well as dependencies and predicted run-times. The DPS provides status and processing completion information to the PLS.

DSS Interface

The PLS queries the DSS inventory for data required for processing. If the data exists, the DSS responds to the PLS with granule information (identification, metadata, and location).

CSS Interface

The CSS Subscription server provides a notification on the arrival of ECS data. The ECS Advertising service provides the advertisement data required by the PLS to generate subscriptions. The PLS exchanges mode management information with and receives event notifications from the CSS.

MSS Interface

The PLS sends fault management, accounting, security, and performance data to the MSS for logging. The PLS receives Order tracking information for On-Demand Products and standing orders.

CLS Interface

The PLS receives requests for On-Demand Products from the CLS (ODFRM) and an Order Id is returned to the user.

Planning Subsystem Structure

The PLS is comprised of one CSCI, Production Planning (PLANG CSCI) and one hardware CI, Production Planning (PLNHW).

The Planning and Data Processing Subsystems (PDPS) database resides in the PLNHW and serves both planning and scheduling activities.

Use of COTS in the Planning Subsystem

- Hughes- Delphi Scheduling Class Libraries.

  The Delphi Scheduling Class Libraries are used to schedule the Resource Planning Workbench and the Production Planning Workbench. Delphi uses C++ classes to provide user-oriented, integrated, and modular planning and scheduling software utilities.

- RogueWave’s Tools.h++

  The Tools.h++ class libraries provide libraries of object strings and collections. These libraries are delivered statically linked with the custom code delivery.

- RogueWave’s DBTools.h++

  The DBTools.h++ C++ class libraries interact with the Sybase ASE database Structured Query Language (SQL) server and buffer the processes from the relational database used. These libraries are delivered statically linked with the custom code delivery.

- ICS’ Builder Xcessory

  The Builder Xcessory GUI builder tool modifies displays. The Builder Xcessory generates the C++ code to produce the Maintenance Tool (Mtool) display at run time. There is no operational component of Builder Xcessory needed at run-time.

- Sybase Adaptive Server Enterprise (ASE)

  The Sybase ASE provides the capabilities to read, insert, update and delete PDPS database content. The Sybase ASE must be operational during the PLS operations.

- CCS Middleware Client

  CCS Middleware Client provides PLS with communications between other subsystems. CCS Middleware can reside on one or both sides of the interface. An instance must be installed on the platform where PLS resides. Although the CCS Middleware Client is part of CSS, this COTS product must be installed for PLS to run in the SDPS operational and test environment.

4.6.1 Production Planning (PLANG) Software Description

4.6.1.1 Production Planning Functional Overview

The PLANG CSCI manages the data production activities at each site by providing the Maintenance and Operations (M&O) staff with the following capabilities:

- Defining the data processing tasks (via data processing requests) to perform at the site.
- Generating data production plans for scheduling processing and reprocessing tasks.
- Coordinating data production with the DSS and the DPS to automate the production system.

The On-Demand Manager (ODPRM) is used to manage the on-demand orders received from the ODFRM. Upon receipt of an order, the On-Demand Manager determines the order type. The order type can be a DEM, an ASTER higher-level product order, or a standing order. Once the order type is determined, the ODPRM verifies that the inputs provided by the ODFRM are valid. The ODPRM uses this information to check on demand production requests, which are then processed by DPS.

If the order is a DEM, the ODPRM simply creates the associated dummy DPRs and granules and returns a MSS order id. Once the request is completed the operator inserts the DEM into the SDSRV and invokes the command line tool again to update the user

If the order is for a higher-level product, the ODPRM creates the PRs and DPRs necessary to fulfill the order. If all inputs are available, the data processing request(s) are submitted to the Data Processing Subsystem and the product is produced. As each DPR changes state, the MSS Order Tracking GUI is updated to reflect the new state and the overall order status is updated accordingly.

If the order is a standing order associated with a DAR, ODPRM stores the product and the DAR information associated with this order in the PDPS database. As granules associated with the DAR arrive, ODPRM automatically generates on demand PRs and DPRs, which follow the same production process as ordinary ASTER high level on demand requests.

4.6.1.2 Production Planning Context

Figure 4.6-2 is the PLANG CSCI context diagrams. The diagrams show the events sent to the PLANG CSCI and the events the PLANG CSCI sends to other CSCIs and the M&O staff. Table 4.6-2 provides descriptions of the interface events shown in the PLANG CSCI context diagrams.

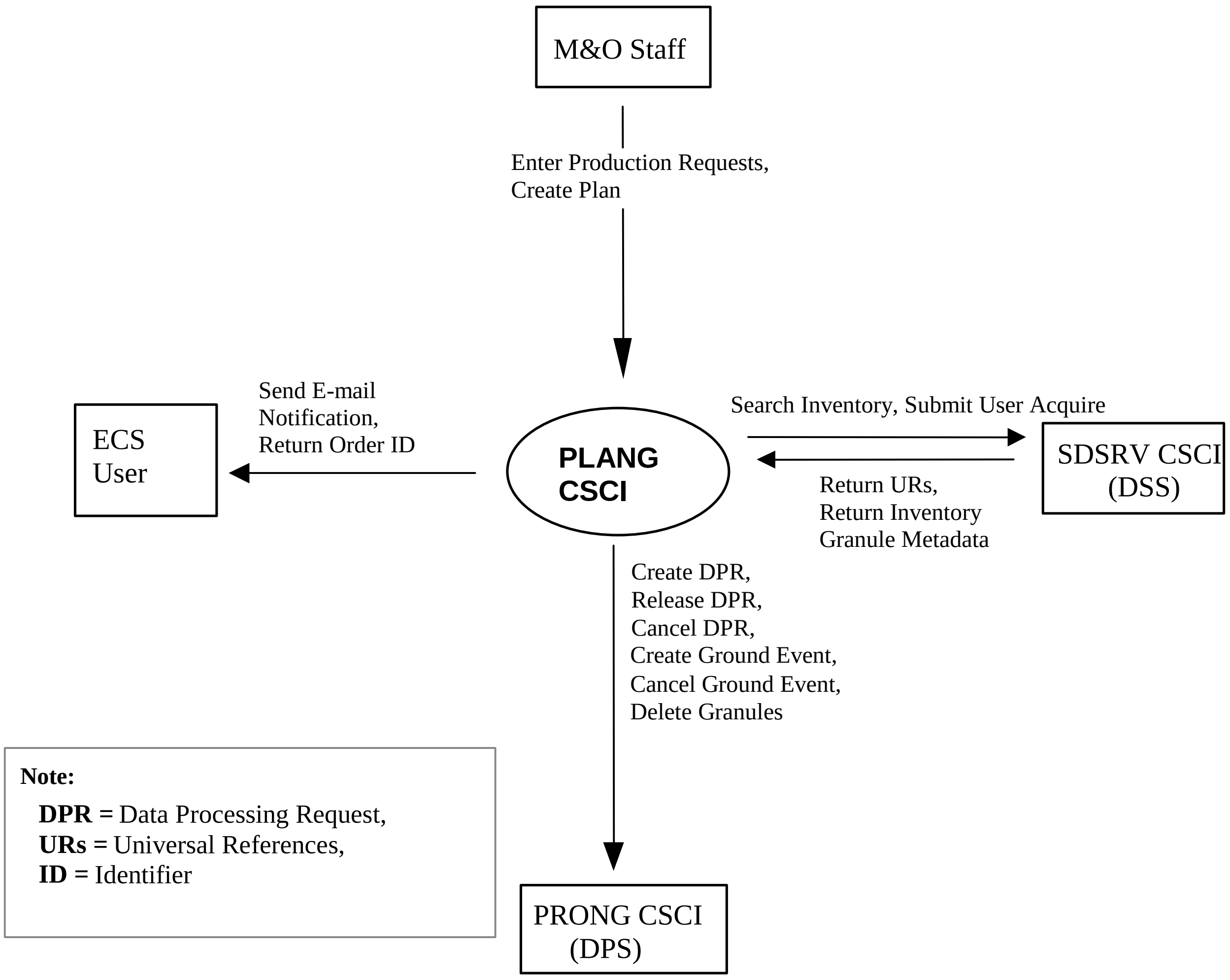

M&O Staff
Enter Production Requests, Create Plan
ECS User
Send E-mail Notification, Return Order ID
PLANG CSCI
Search Inventory, Submit User Acquire
SDSRV CSCI (DSS)
Return URs, Return Inventory Granule Metadata
Create DPR, Release DPR, Cancel DPR, Create Ground Event, Cancel Ground Event, Delete Granules
Note:
DPR = Data Processing Request,
URs = Universal References,
ID = Identifier
PRONG CSCI (DPS)

Figure 4.6-2. PLANG CSCI Context Diagram

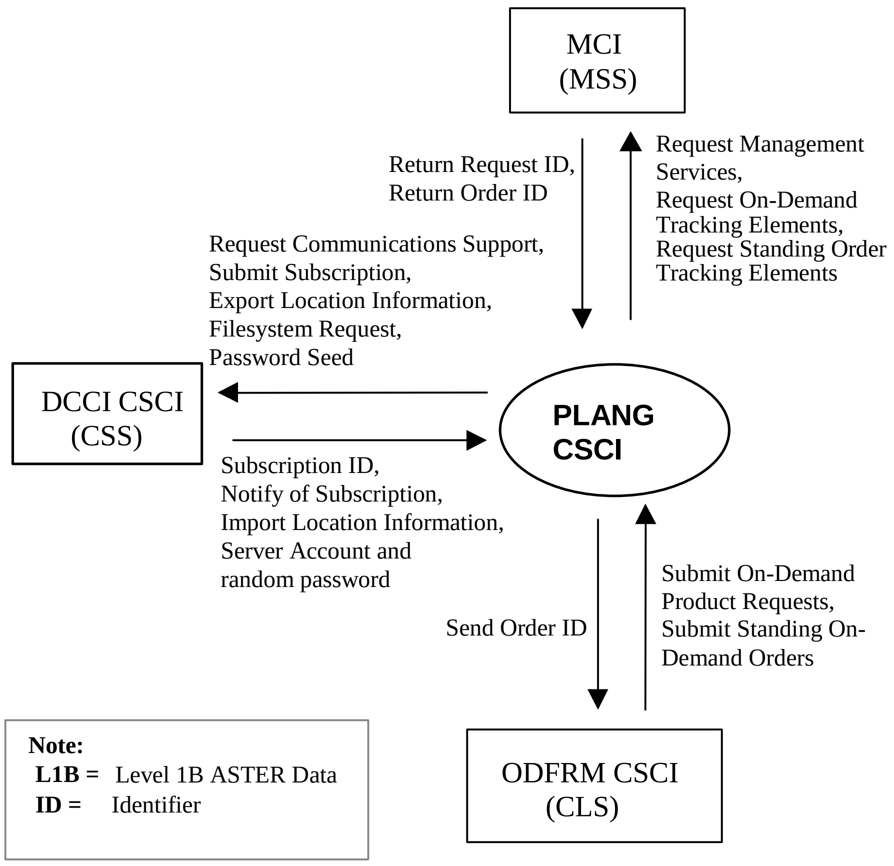

MCI
(MSS)
Return Request ID,
Return Order ID
Request Management Services,
Request On-Demand Tracking Elements,
Request Standing Order Tracking Elements
Request Communications Support,
Submit Subscription,
Export Location Information,
Filesystem Request,
Password Seed
DCCI CSCI
(CSS)
PLANG CSCI
Subscription ID,
Notify of Subscription,
Import Location Information,
Server Account and random password
Send Order ID
Submit On-Demand Product Requests,
Submit Standing On-Demand Orders
Note:
L1B = Level 1B ASTER Data
ID = Identifier
ODFRM CSCI
(CLS)

Figure 4.6-2. PLANG CSCI Context Diagram (cont.)

Table 4.6-2. PLANG CSCI Interface Events (1 of 3)

Event	Interface Event Description
Enter Production Requests	The **M&O staff** request production by selecting a PGE type and the time duration for the PGE to process the input data in the PLANG CSCI.
Create Plan	The **M&O staff** issues commands to control the creation and activation of a data production plan to the PLANG CSCI.
Search Inventory	The PLANG CSCI sends inventory search or inspect requests to the **SDSRV CSCI** to search the ECS inventory/archives (granules). In response, the PLANG CSCI receives granule URs satisfying the search.
Submit User Acquire	The PLANG CSCI submits an acquire command to the **SDSRV CSCI** on behalf of the user. The user gets a response via the DDIST CSCI upon data distribution.
Return URs	The PLANG CSCI receives Earth Science Data Type (ESDT) Universal References (URs) for the granules from the **SDSRV CSCI**.

Table 4.6-2. PLANG CSCI Interface Events (2 of 3)

Event	Interface Event Description
Return Inventory Granule Metadata	The PLANG CSCI receives the inventory granule metadata identifying the scene within the granule based on an inventory search request sent to the **SDSRV CSCI**.
Create DPR	The PLANG CSCI sends the Data Processing Request Identification (dprId) and whether the DPR is waiting for external input data to the **PRONG CSCI**.
Release DPR	The PLANG CSCI sends the dprId to the **PRONG CSCI**.
Cancel DPR	The PLANG CSCI sends a request to cancel the dprId to the **PRONG CSCI** for the deletion of a DPR.
Create Ground Event	The PLANG CSCI sends the ground event id, resource id, and start time to the **PRONG CSCI** to create a ground event to perform maintenance activities on data processing resources.
Cancel Ground Event	The PLANG CSCI sends the ground event id, resource id, and start time to the **PRONG CSCI** to delete a ground event.
Delete Granules	The PLANG CSCI sends requests to the **PRONG CSCI** to delete granules associated with cancelled DPRs.
Send E-mail Notification	The PLANG CSCI sends e-mail notification to the **user** when a standing order granule arrives and when a standing order expires.
Return Order ID	The PLANG CSCI sends an order id to the **user** to track his on-demand product or order requests.
Request Management Services	**System startup and shutdown -** Please refer to the release-related, current version of the Mission Operations Procedures for the ECS Project document (611) and the current ECS Project Training Material document (625), identified in Section 2.2.1 of this document.
Request On-Demand Tracking Elements	The PLANG CSCI sends product requests to the **MCI** for orders.
Request Standing Order Tracking Elements	The PLANG CSCI sends requests to the **MCI** for standing orders of ASTER high-level products.
Submit On-Demand Product Requests	The PLANG CSCI receives on-demand requests from the **ODFRM CSCI**. As a result, the user receives an Order ID. The user receives a notification when the request is processed.
Submit Standing On-Demand Orders	The PLANG CSCI receives standing on-demand orders from the **ODFRM CSCI**. As a result, the user receives a standing order ID.
Send Order ID	The PLANG CSCI returns the order identifier to the **ODFRM CSCI**.
Subscription ID	The PLANG CSCI receives a subscription identifier from the **DCCI CSCI** after submitting a subscription.
Notify of Subscription	A message passing callback in the PLANG CSCI subscription manager is called, by the **DCCI CSCI**, with the granule UR inserted into the SDSRV inventory as a calling parameter.
Import Location Information	The PLANG CSCI requests server location information from the **DCCI CSCI**.
Server Account and random password	The PLANG CSCI receives an account and random password from the **DCCI CSCI** after providing a password seed to establish an account.

Table 4.6-2. PLANG CSCI Interface Events (3 of 3)

Event	Interface Event Description
Request Communications Support	The **DCCI CSCI** provides a library of services available to each SDPS and CSMS CSCI. The CSCI services required to perform specific assignments are requested from the DCCI CSCI. These services include: • CCS Middleware Support • Database Connection Services • Network & Distributed File Services • Name/Address Services • Password Services • Server Request Framework (SRF) • Universal Reference (UR) • Error/Event Logging • Message Passing • Fault Handling Services • Mode Information • Query Registry - Retrieving the requested configuration attribute-value pairs from the Configuration Registry
Submit Subscription	The PLANG CSCI submits a subscription for an ESDT insert event to the **DCCI CSCI**. In response, the PLANG CSCI receives a subscription identifier.
Export Location Information	The PLANG CSCI stores physical and logical server location information in the **DCCI CSCI**.
Filesystem Request	The PLANG CSCI requests ECS files and directories via an established mount point from the **DCCI CSCI**. The DCCI CSCI makes the storage device(s) and its data accessible for use by clients.
Password Seed	The PLANG CSCI requests an account and provides a password to the **DCCI CSCI**.

PLANG CSCI interfaces include:

PDPS Database Interface (Common database pseudo-interface with DPS)

The DPS Algorithm Integration and Test Tools (AITTL) CSCI stores PGE data, which is retrieved by the PLS. This PGE data includes the PGE executable, the input data type(s) it requires, the output data type(s) it generates, and the resource requirements (e.g., hardware platform, memory, and disk storage). The PGE data is used by the PLS to schedule data processing requests with the DPS.

The PLS manages the database space by deleting DPRs from the PDPS database and some of its associated granules not used by other DPRs.

Operator Interface

The Maintenance and Operations (M&O) staff personnel enter Production Requests into the PLS via the Planning User Interface. Production Requests provide the information necessary for data to be produced by the DPS. Production Requests are used to process new data (Routine Production Requests) or for reprocessing data (Reprocessing Production Requests). The PLS uses the PGE profile information from the Production Requests to generate the DPRs needed to fulfill the request for data. The Planning User Interface also issues commands to initiate plan creation, plan activation and plan cancellations, and provide reports and status of plan progress. The M&O staff performs resource planning for the entire DAAC through the Planning User Interface with awareness of the impact of ground events on data processing resources.

4.6.1.3 Production Planning Architecture

Figure 4.6-3 is the PLANG CSCI architecture diagrams without the On-Demand Manager included. The diagrams show the events sent to the PLANG CSCI processes and the events sent by the PLANG CSCI processes to other processes. Figure 4.6-3 is the PLANG CSCI architecture diagrams with the On-Demand Manager featured. The diagrams show the events sent to the PLANG CSCI processes and the events sent by the PLANG CSCI processes to other processes.

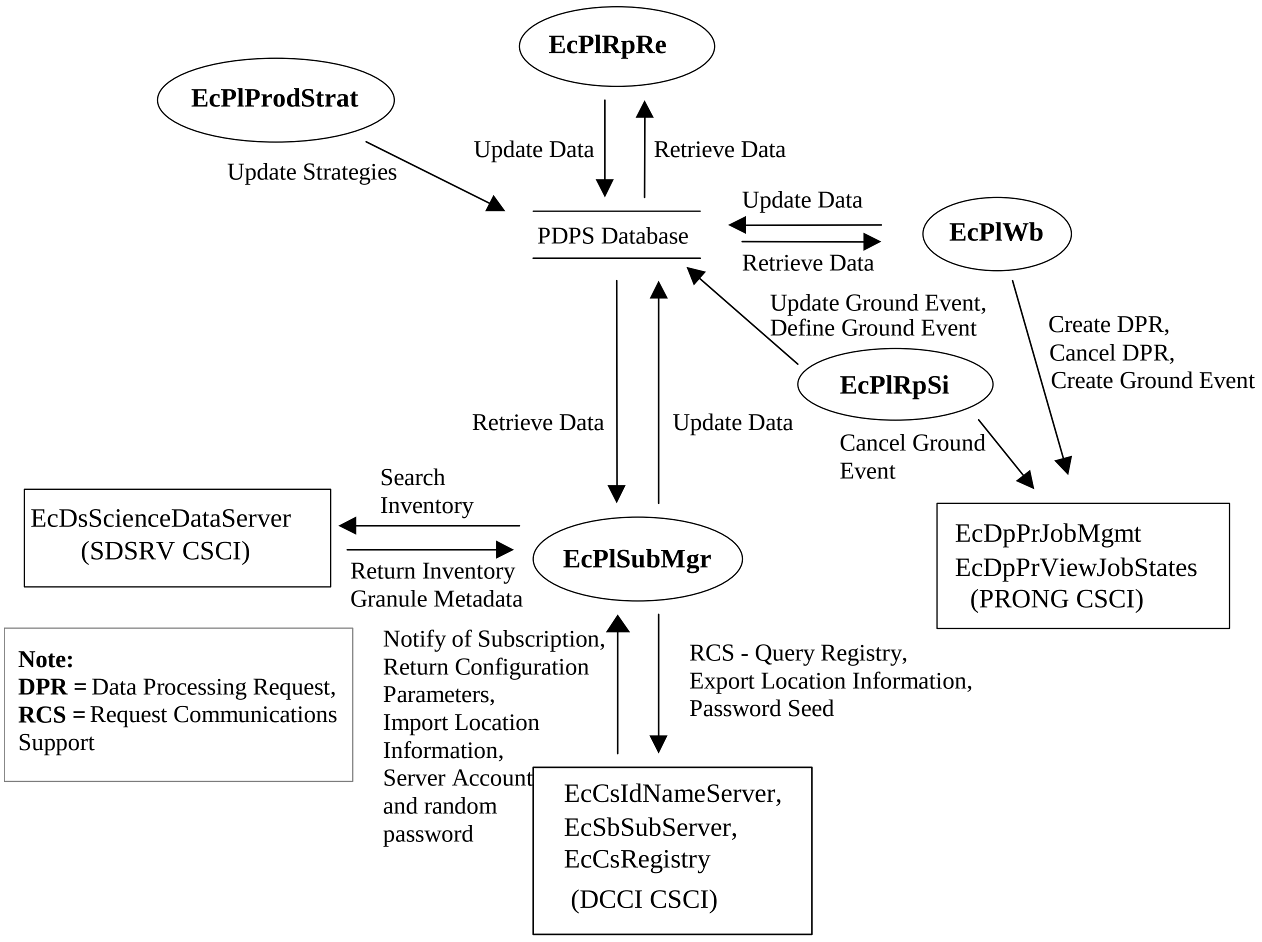


Figure 4.6-3. PLANG CSCI Architecture Diagram

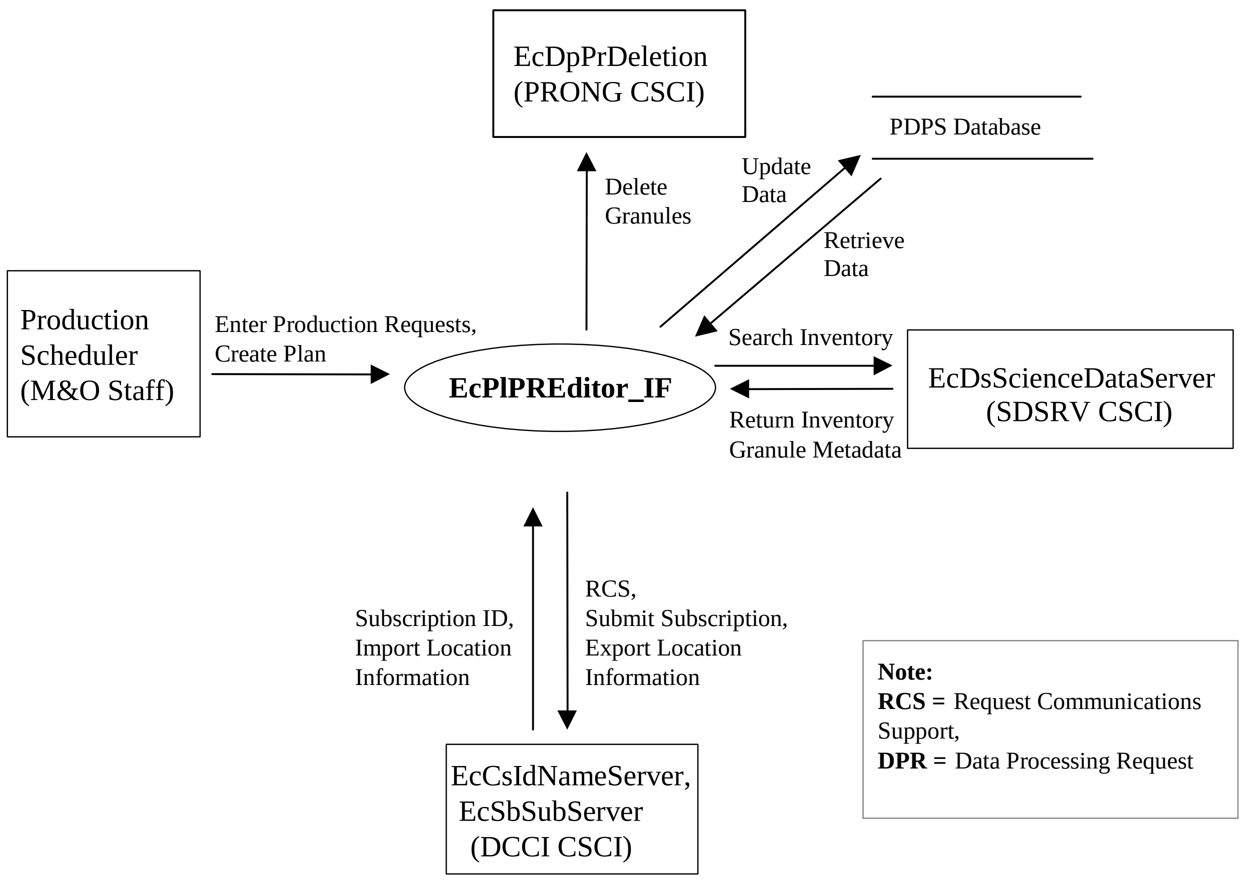

EcDpPrDeletion (PRONG CSCI)
PDPS Database
Delete Granules
Update Data
Retrieve Data
Production Scheduler (M&O Staff)
Enter Production Requests, Create Plan
EcPlPREditor_IF
Search Inventory
Return Inventory Granule Metadata
EcDsScienceDataServer (SDSRV CSCI)
Subscription ID, Import Location Information
RCS, Submit Subscription, Export Location Information
EcCsIdNameServer, EcSbSubServer (DCCI CSCI)
Note:
RCS = Request Communications Support,
DPR = Data Processing Request

Figure 4.6-3. PLANG CSCI Architecture Diagram (cont.)

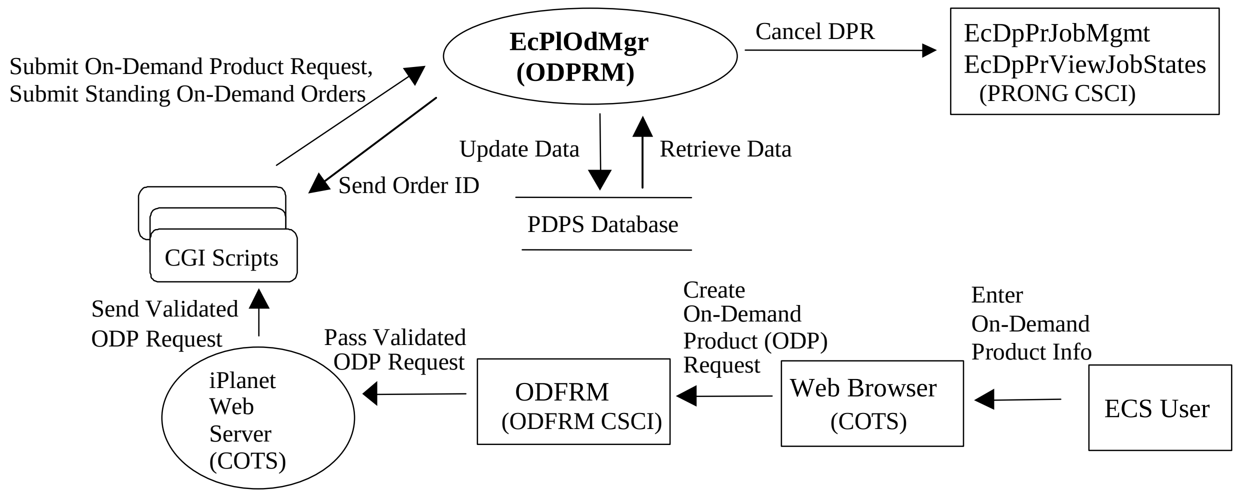

EcPlOdMgr (ODPRM)
Cancel DPR
EcDpPrJobMgmt EcDpPrViewJobStates (PRONG CSCI)
Submit On-Demand Product Request, Submit Standing On-Demand Orders
Send Order ID
Update Data
Retrieve Data
PDPS Database
CGI Scripts
Send Validated ODP Request
Pass Validated ODP Request
Create On-Demand Product (ODP) Request
Enter On-Demand Product Info
iPlanet Web Server (COTS)
ODFRM (ODFRM CSCI)
Web Browser (COTS)
ECS User

Figure 4.6-3. PLANG CSCI Architecture Diagram (cont.)

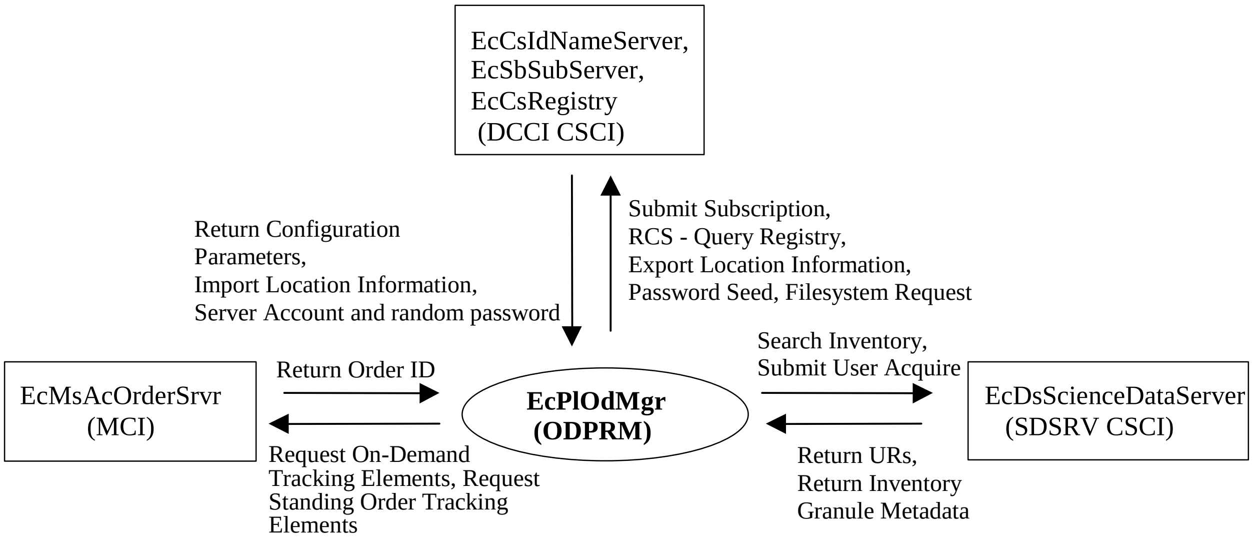

EcCsIdNameServer,
EcSbSubServer,
EcCsRegistry
(DCCI CSCI)
Return Configuration Parameters,
Import Location Information,
Server Account and random password
Submit Subscription,
RCS - Query Registry,
Export Location Information,
Password Seed, Filesystem Request
EcMsAcOrderSrvr
(MCI)
Return Order ID
Request On-Demand Tracking Elements, Request Standing Order Tracking Elements
EcPlOdMgr
(ODPRM)
Search Inventory,
Submit User Acquire
Return URs,
Return Inventory Granule Metadata
EcDsScienceDataServer
(SDSRV CSCI)

Figure 4.6-3. PLANG CSCI Architecture Diagram (cont.)

4.6.1.4 Production Planning Process Descriptions

Table 4.6-3 provides descriptions of the Production Planning processes shown in the PLANG CSCI architecture diagrams.

Table 4.6-3. PLANG CSCI Processes (1 of 2)

Process	Type	COTS / Developed	Functionality
EcPlRpRe	GUI	Developed code using Delphi Class Libraries.	The Resource Planning Workbench prepares a schedule for the resources at each respective site, and forecasts the start and completion times of the ground events and the impact on the resources used within the schedule. The workbench allows the M&O staff to: • Edit the resources currently available at a site • Associate the resources with production strings (logical groupings of resources used by AutoSys and Data Processing) when allocating resources for a particular PGE.
EcPlWb	GUI	Developed code using Delphi Class Libraries.	The Production Planning Workbench prepares a schedule for the production at a site, and forecasts the start and completion times of the activities within the schedule. Specifically, the Workbench allows: • Candidate Plan Creation—from the production requests prepared by the Production Request Editor • Plan Activation—activating a candidate plan • Update of the Active Plan—feedback from the DPS activities are incorporated into the active plan • Cancellation/Modification of the Active Plan Activating a plan entails rolling a portion of a selected plan into the AutoSys COTS via the DPS. The “schedule” is managed within the DPS. The forecast times generated by the planner are used to set up operator alerts to gross departures from the predicted schedule. Ground Events are sent to the DPS via the EcPlWb.
EcPlPREditor_IF	GUI	Developed	The Planning User Interface (Production Request Editor) allows the M&O staff to submit production requests to describing the data products to generate. The production request uses the PGE descriptions (profiles) entered during Algorithm Integration and Test (AI&T) to define the Data Processing Requests. The request adds, modifies, and deletes Production Requests, and reviews and modifies the resulting Data Processing Requests. The user specifies rules for producing the individual DPRs for the reprocessing requests. The production request editor is a distinct application and separate from the workbench because defining a production request is unrelated to the planning of a production request.
EcPlRpSi	GUI	Developed code using Delphi Class Libraries	The Planning Resource GUI allows the M&O staff to: • Define or cancel ground events (maintenance, etc.) on the allocated resources

Table 4.6-3. PLANG CSCI Processes (2 of 2)

Process	Type	COTS / Developed	Functionality
EcPlSubMgr	Server	Developed	The Subscription Manager receives subscription notifications from the EcDsScienceDataServer via the EcSbSubServer. Subscription notification notifies planning of the arrival of required input data. The Subscription Notification is handled through the Infrastructure message passing service and contains URs pointing to the data objects stored in the EcDsScienceDataServer. The Subscription Manager updates the PDPS database when data is available. When all input data for a DPR is available, the job defined for that DPR is released within the DPS.
EcPlProdStrat	GUI	Developed	The Production Strategies GUI is used to create a set of planning priorities to be applied to each DPR in a plan. This strategy takes user, PGE type, PGE instance, and Production Request priorities into account. This strategy is then saved to the PDPS database.
EcPlOdMgr	Server	Developed	The On-Demand manager (ODPRM) receives requests for data from the scientist via the ODFRM web page. The scientist can request a DEM, non-standard L1B or a higher-level product and attached DPR standing orders. Once the ODPRM has generated the Production Request (PR) necessary to fulfill the request, in the case of the higher-level product, it submits the PR to DPS for processing. Once DPS is finished, the ODPRM distributes the data to the scientist who requested it.

4.6.1.5 Production Planning Process Interface Descriptions

Table 4.6-4 provides descriptions of the interface events shown in the PLANG CSCI architecture diagrams without the On-Demand Manager included. Table 4.6-4 provides descriptions of the interface events shown in the PLANG CSCI architecture diagrams with the On-Demand Manager featured (starting at table 12 of 19).

Table 4.6-4. PLANG CSCI Process Interface Events (1 of 19)

Event	Event Frequency	Interface	Initiated By	Event Description
Retrieve Data	One per retrieve request	PDPS Database	***Process:*** EcPlRpRe *Library:* PlRpRe *Classes:* PlRpReAutosysWin, PlRpReComputerWin, PlRpReDiskWin, PlRpReHardwareWin, PlRpReRealComputer Win, PlRpReStringWin, PlRpReWin ***Process:*** EcPlWb *Library:* plwb *Class:* PlWbScheduler **Process:** EcPlSubMgr Library: PlCore1 Classes: PlDataGranule, PlProductionRequest	The EcPlRpRe, EcPlWb and EcPlSubMgr processes send requests to the **PDPS database** to retrieve information to allow the PGE to be scheduled and executed.

Table 4.6-4. PLANG CSCI Process Interface Events (2 of 19)

Event	Event Frequency	Interface	Initiated By	Event Description
Update Data	One update per request	PDPS Database	***Process:*** EcPlWb *Library:* plwb *Class:* PlWbScheduler **Process:** EcPlSubMgr Library: PlCore1 Classes: PlDataGranule, PlProductionRequest ***Process:*** EcPlRpRe *Library:* PlRpRe *Classes:* PlRpReAutosysWin, PlRpReComputerWin, PlRpReDiskWin, PlRpReHardwareWin, PlRpReRealComputer Win, PlRpReStringWin, PlRpReWin	The EcPlWb, EcPlSubMgr and EcPlRpRe processes send requests to the **PDPS database** to update granule information (location, size, etc.), processing status and check pointing.
Update Ground Event	Once per defined ground event	PDPS Database	*Process:* EcPlRpSi *Library:* PlRpSi *Classes:* PlRpSiScheduler, PlRpSiWinAbs	The EcPlRpSi also updates ground event information in the **PDPS database**.

Table 4.6-4. PLANG CSCI Process Interface Events (3 of 19)

Event	Event Frequency	Interface	Initiated By	Event Description
Define Ground Event	One per request	PDPS Database	*Process:* EcPlRpSi *Library:* PlRpSi *Classes:* PlRpSiScheduler, PlRpSiWinAbs	The EcPlRpSi provides the information to identify the ground events to be done and sends the information to the **PDPS database**.
Create DPR	One per list of predecessor DPRs	*Process:* EcDpPrJobMgmt *Library:* DpPrJM *Class:* DpPrScheduler	*Process:* EcPlWb *Class:* PlWbScheduler	The EcPlWb process sends the dprId and whether the DPR is waiting for external data to the **EcDpPrJobMgmt** process to create a job in the data production process.
Cancel DPR	One per dprId send	*Process:* EcDpPrJobMgmt *Library:* DpPrJM *Class:* DpPrScheduler	*Process:* EcPlWb *Library:* PlCore1 *Class:* PlDpr	The EcPlWb process sends a request to cancel the dprId to the **EcDpPrJobMgmt** process for the deletion of a DPR.
Create Ground Event	One per defined ground event	*Process:* EcDpPrJobMgmt *Library:* DpPrJM *Class:* DpPrScheduler	*Process:* EcPlWb *Library:* plwb *Class:* PlWbScheduler	The EcPlWb process sends the ground event id, resource id, and start time to the **EcDpPrJobMgmt** process to perform maintenance activities on data processing resources.
Cancel Ground Event	One per defined ground event	*Process:* EcDpPrJobMgmt *Library:* DpPrJM *Class:* DpPrScheduler	*Process:* EcPlRpSi *Library:* PlRpSi *Classes:* PlRpSiScheduler, PlRpSiWinAbs	The EcPlRpSi sends a request to cancel a ground event to the **EcDpPrJobMgmt** process for the deletion of a ground event.

Table 4.6-4. PLANG CSCI Process Interface Events (4 of 19)

Event	Event Frequency	Interface	Initiated By	Event Description
Request Communications Support	One per process request.	***Process:*** EcCsIdNameServer ***Libraries:*** EcPf, Middleware, FoNs, FoIp, oodce *Classes:* EcPfManagedServer, EcPfClient, CCSMdwNameServer, FoNsNameServerProxy, CCSMdwRwNetProxy ***Library (Common):*** EcUr *Class:* EcUrServerUR ***Process:*** EcSbSubServer *Library:* EcSbSr *Class*: EcSbSubscription ***Library:*** EcDcMsgPsng1 ***Library:*** event *Class:* EcLgErrorMsg ***Library:*** EcSeCmi Class: EcSeCmi ***Process:*** EcCsRegistry *Library:* EcCsRegistry *Class:* EcRgRegistryServer_C	*Process:* EcPlSubMgr *Library:* PlCore1 *Classes:* Most PLS Classes	The **DCCI CSCI** provides a library of services available to each SDPS and CSMS CSCI. The CSCI services required to perform specific assignments are requested from the DCCI CSCI. These services include: • CCS Middleware Support • Database Connection Services • Network & Distributed File Services • Name/Address Services • Password Services • Server Request Framework (SRF) • Universal Reference (UR) • Error/Event Logging • Message Passing • Fault Handling Services • Mode Information • Query Registry - Retrieving the requested configuration attribute-value pairs from the Configuration Registry

Table 4.6-4. PLANG CSCI Process Interface Events (5 of 19)

Event	Event Frequency	Interface	Initiated By	Event Description
Export Location Information	Once at system startup and after each failure recovery	*Process:* EcCsIdNameServer *Libraries:* EcPf, Middleware, FoNs, FoIp, oodce *Classes:* EcPfManagedServer, EcPfClient, CCSMdwNameServer, FoNsNameServerProxy, CCSMdwRwNetProxy	*Process:* EcPlSubMgr *Library:* PlCore1 PlCore1IF PlCore1Sub PlCore2 PlSSIT PlUtil *Classes:* PlSubscriptionManager Process.C, PlSubscriptionManager .C	The EcPlSubMgr stores physical and logical location information in the **EcCsIdNameServer**.
Password Seed	One per password seed	*Process:* Cryptographic Management Interface *Library:* EcSeCmi *Class:* EcSeCmi	*Processes:* EcPlSubMgr, EcPlRpRe, EcPlRpRm, EcPlRpSi, EcPlRpT1, EcPlWb	The server provides a unique number as a seed for generating a password to the **EcPlSubMgr,** EcPlRpRe, EcPlRpRm, EcPlRpSi, EcPlRpT1 **and EcPlWb** processes.
Notify of Subscription	One per message passing callback	*Process:* EcPlSubMgr *Class:* PlSubMsgCb	*Process:* EcSbSubServer *Library:* EcSbSr *Class:* EcSbSubscription	The **EcSbSubServer** calls a message passing callback in the Subscription Service, with the granule UR inserted into the data server as a calling parameter, to send notification of a subscription event to the EcPlSubMgr.
Return Configuration Parameters	One set per request	*Process:* EcPlSubMgr *Library:* PlCore1 PlCore1IF PlCore1Sub PlCore2 PlSSIT PlUtil *Classes:* PlSubscriptionManagerP rocess.c, PlSubscriptionManager. C	*Process:* EcCsRegistry *Library:* EcCsRegistry Class: EcRgRegistryServer_C	The **EcCsRegistry** returns the attribute-value pairs (configuration parameters) to the EcPlSubMgr upon request.

Table 4.6-4. PLANG CSCI Process Interface Events (6 of 19)

Event	Event Frequency	Interface	Initiated By	Event Description
Import Location Information	As required for processing	*Process:* EcPlSubMgr *Library:* PlCore1 PlCore1IF PlCore1Sub PlCore2 PlSSIT PlUtil *Classes:* PlSubscriptionManagerProcess.c, PlSubscriptionManager.C	*Process:* EcCsIdNameServer *Libraries:* EcPf, Middleware, FoNs, FoIp, oodce *Classes:* EcPfManagedServer, EcPfClient, CCSMdwNameServer, FoNsNameServerProxy, CCSMdwRwNetProxy	The EcPlSubMgr requests server location information from the **EcCsIdNameServer**.
Server Account and random password	One per account and password	*Processes:* EcPlSubMgr, EcPlRpRe, EcPlRpRm, EcPlRpSi, EcPlRpT1, EcPlWb *Library:* PlCore1 PlCore1IF PlCore1Sub PlCore2 PlSSIT PlUtil *Classes:* PlSubscriptionManagerProcess.c, PlSubscriptionManager.C	*Process:* Cryptographic Management Interface *Library:* EcSeCmi *Class:* EcSeCmi	The Cryptographic Management Interface generates a random password for the account based on the seed provided by the EcPlSubMgr, and EcPlRpRe, EcPlRpRm, EcPlRpSi, and EcPlRpT1 processes.

Table 4.6-4. PLANG CSCI Process Interface Events (7 of 19)

Event	Event Frequency	Interface	Initiated By	Event Description
Search Inventory	One per query	*Process:* EcDsScienceDataServer *Library:* DsCl *Class:* DsClQuery	*Process:* EcPlSubMgr *Library:* DpPrDssIF *Class:* DpPrDSSInterface	The EcPlSubMgr process creates two types of queries. One type only has the ESDT short name and data start and stop times and the other type also includes spatial coordinates. The EcPlSubMgr process creates an ESDT Reference from an UR after receiving a subscription notification or receiving an ESDT reference from a query. The EcPlSubMgr process queries when predicted data is not available. The **EcDsScienceDataServer** returns metadata information about the granule being inspected.
Return Inventory Granule Metadata	All metadata per inventory search	*Process:* EcPlSubMgr *Library:* DpPrDssIF *Class:* DpPrDSSInterface	*Process:* EcDsScienceDataServer *Library:* DsCl *Class:* DsClQuery	The EcPlSubMgr receives the inventory granule metadata identifying the scene within the granule based on an inventory search request sent to the **EcDsScienceDataServer**.
Update Strategies	One per strategy created.	Active strategies screen	M&O staff *Process:* EcPlProdStrat *Library:* PlGUI *Class:* PlProdStratActive	The **M&O staff** creates strategies via the EcPlProdStrat process when certain jobs need to be prioritized over others. The strategy is saved by name and later can be read by the EcPlWb to prioritize the DPRs in a plan.

Table 4.6-4. PLANG CSCI Process Interface Events (8 of 19)

Event	**Event Frequency**	**Interface**	**Initiated By**	**Event Description**
Delete Granules	One set of granules per cancelled DPR	*Process:* EcDpPrDeletion *Class:* DpDeletion	*Process:* EcPlPREditor_IF *Library:* PlCore1 *Classes:* PlWbScheduler, PlDpr	The EcPlPREditor_IF sends requests to the **EcDpPrDeletion** process to delete granules associated with a DPR, which has been cancelled.
Update Data	One per update/retrieve request	PDPS Database	*Process:* EcPlPREditor_IF *Library:* PlCore1 *Class:* PlDataType	Requests are sent for updates of granule information (location, size, etc.), processing status, and check pointing from the EcPlPREditor_IF to the **PDPS Database**.
Retrieve Data	One per retrieve request	PDPS Database	*Process:* EcPlPREditor_IF *Library:* PlCore1 *Class:* PlDataType	The EcPlPREditor_IF processes send requests to the **PDPS database** to update/retrieve data defining a PGE. Also, the requests contain information to allow the PGE to be scheduled and executed.
Search Inventory	One per query	*Process:* EcDsScienceDataServer *Library:* DsCl *Class:* DsClQuery	*Process:* EcPlPREditor_IF *Library:* DpPrDssIF *Class:* DpPrDSSInterface	The EcPlPREditor_IF process creates two types of queries. One type only has the ESDT short name and data start and stop times and the other type also includes spatial coordinates. The EcPlPREditor_IF process queries when the predicted data is available. The EcPlPREditor_IF process creates an ESDT Reference from an UR after receiving an ESDT Reference from a query. The **EcDsScienceDataServer** returns ESDT References for granules to satisfy the query.
Return Inventory Granule Metadata	All metadata per inventory search	*Process:* EcPlPREditor_IF *Library:* DpPrDssIF *Class:* DpPrDSSInterface	*Process:* EcDsScienceDataServer *Library:* DsCl *Class:* DsClQuery	The EcPlPREditor_IF receives the inventory granule metadata identifying the scene within the granule based on an inventory search request sent to the **EcDsScienceDataServer**.

Table 4.6-4. PLANG CSCI Process Interface Events (9 of 19)

Event	Event Frequency	Interface	Initiated By	Event Description
Request Communications Support	One per process request.	***Process:*** EcCsIdNameServer ***Libraries:*** EcPf, Middleware, FoNs, FoIp, oodce *Classes:* EcPfManagedServer, EcPfClient, CCSMdwNameServer, FoNsNameServerProxy, CCSMdwRwNetProxy ***Library (Common):*** EcUr *Class:* EcUrServerUR ***Process:*** EcSbSubServer *Library:* EcSbSr *Class*: EcSbSubscription ***Library:*** EcDcMsgPsng1 ***Library:*** event *Class:* EcLgErrorMsg ***Library:*** EcSeCmi Class: EcSeCmi ***Process:*** EcCsRegistry *Library:* EcCsRegistry *Class:* EcRgRegistryServer_C	*Process:* EcPlPREditor_IF *Library:* PREGUI PlCore1 PlCore1IF PlCore1Sub PlCore2 PlSSIT PlUtil *Classes:* main_c.cxx, PlPRapp.cxx, PlProdReqEdit.c xx	The **DCCI CSCI** provides a library of services available to each SDPS and CSMS CSCI. The CSCI services required to perform specific assignments are requested from the DCCI CSCI. These services include: • CCS Middleware Support • Database Connection Services • Network & Distributed File Services • Name/Address Services • Password Services • Server Request Framework (SRF) • Universal Reference (UR) • Error/Event Logging • Message Passing • Fault Handling Services • Mode Information • Query Registry - Retrieving the requested configuration attribute-value pairs from the Configuration Registry

Table 4.6-4. PLANG CSCI Process Interface Events (10 of 19)

Event	Event Frequency	Interface	Initiated By	Event Description
Submit Subscription	One per subscription created	*Process:* EcSbSubServer *Library:* EcSbSr *Class:* EcSbSubscription	*Process:* EcPlPREditor *Library*: PlCore1 *Class*: PlDataType	The EcPlPREditor_IF process creates subscriptions using the advertisement for subscribing to an ESDT insert event and enters the subscriptions via the **EcSbSubServer**.
Export Location Information	Once at system startup and after each failure recovery	*Process:* EcCsIdNameServer *Libraries:* EcPf, Middleware, FoNs, FoIp, oodce *Classes:* EcPfManagedServer, EcPfClient, CCSMdwNameServer, FoNsNameServerProxy, CCSMdwRwNetProxy	*Process:* EcPlPREditor_IF *Library:* PREGUI PlCore1 PlCore1IF PlCore1Sub PlCore2 PlSSIT PiUtil *Classes:* main_c.cxx, PlPRapp.cxx, PlProdReqEdit.cxx	The EcPlPREditor_IF stores physical and logical location information in the **EcCsIdNameServer**.
Subscription ID	One per subscription request	*Process:* EcPlPREditor *Library*: PlCore1 *Class*: PlDataType	*Process:* EcSbSubServer *Library:* EcSbSr *Class:* EcSbSubscription	The EcPlPREditor_IF receives a subscription identifier from the **EcSbSubServer** after submitting a subscription.

Table 4.6-4. PLANG CSCI Process Interface Events (11 of 19)

Event	Event Frequency	Interface	Initiated By	Event Description
Import Location Information	As required for processing	*Process:* EcPlPREditor_IF *Library:* PREGUI PlCore1 PlCore1IF PlCore1Sub PlCore2 PlSSIT PiUtil *Classes:* main_c.cxx, PlPRapp.cxx, PlProdReqEdit.cxx	*Process:* EcCsIdNameServer *Libraries:* EcPf, Middleware, FoNs, FoIp, oodce *Classes:* EcPfManagedServer, EcPfClient, CCSMdwNameServer, FoNsNameServerProxy, CCSMdwRwNetProxy	The EcPlPREditor_IF requests server location information from the **EcCsIdNameServer**.
Enter Production Requests	One per production request	*Process:* EcPlPREditor *Library:* DpPrDssIF *Class:* DpPrDSSInterface	M&O staff	The **M&O staff** request production by selecting a PGE type and the time duration for the PGE to process the input data.
Create Plan	One per created and activated plan	*Process:* EcPlWb *Library:* plwb *Class:* PlWbScheduler	M&O staff	The **M&O staff** creates and activates a data production plan.
Request Management Services	At system startup or shutdown and for restarts	*Processes:* EcPlRpRe, EcPlSubMgr	DAAC unique startup scripts	**System startup and shutdown -** Please refer to the release-related, current version of the Mission Operations Procedures for the ECS Project document (611) and the current ECS Project Training Material document (625), identified in Section 2.2.1 of this document.

Table 4.6-4. PLANG CSCI Process Interface Events (12 of 19)

Event	Event Frequency	Interface	Initiated By	Event Description
Cancel DPR	One per list of predecessor DPRs	*Process:* EcDpPrJobMgmt *Library:* DpPrJM *Class:* DpPrScheduler	*Process:* EcPlOdMgr *Class:* PlPrActivator	The EcPlOdMgr process sends a request to cancel the dprId to the **EcDpPrJobMgmt** process for the deletion of a DPR.
Retrieve Data	One per update/retrieve request	*Process:* EcPlOdMgr *Libraries:* PlCore1, PlOdMgrClient *Class:* PlDataType	PDPS Database	The EcPlOdMgr process receives from the **PDPS database** data defining a PGE and information to allow the PGE to be scheduled and executed.
Enter On-Demand Product Info	One per user request	*Process:* Netscape Communicator Web Browser (COTS)	User	The user fills in the User Information on the Login screen and presses the submit button.
Create On-Demand Product (ODP) Requests	One per user request	*Process:* ODFRM	Process: Netscape Communicator Web Browser (COTS)	The ODFRM receives the On-Demand product information and validates the information.
Pass Validated ODP Request	One per user request	*Process:* iPlanet Web Server (COTS)	*Process:* ODFRM	The **iPlanet Web Server** spawns the process EcClOdRequest with the User Login information (name and password).
Send Validated ODP Request	One per user request	CGI Interface, *Process:* EcClOdProductRequest *Library:* ClOdCommon *Class:* ClOdProductRequest	iPlanet Web Server (COTS) *Class:* PlOrderFactory	The EcClOdProductRequest process accesses the MSS database and sends the user back the Authentication.

Table 4.6-4. PLANG CSCI Process Interface Events (13 of 19)

Event	Event Frequency	Interface	Initiated By	Event Description
Submit On-Demand Product Request	One per user request	*Process:* EcPlOdMgr *Library:* PlOdMgrClient *Class:* PlOdMsgProxy	CGI Interface, *Process:* EcClOdProductRequest *Library:* ClOdCommon *Class:* ClOdProductRequest	The **EcClOdProductRequest** process causes an order to be created in the MSS database and sends the request to the EcPlOdMgr, which sends the user back the Order ID.
Submit Standing On-Demand Orders	One per user request	*Process:* EcPlOdMgr *Library:* PlOdMgrClient *Class:* PlOdMsgProxy	CGI Interface, *Process:* EcClOdProductRequest *Library:* ClOdCommon *Class:* ClOdProductRequest	The **EcClOdProductRequest** process causes an order to be created in the MSS database and sends the request to the EcPlOdMgr, which sends the user back the Order ID.
Send Order ID	One per on-demand product request or standing order	CGI Interface, *Process:* EcClOdProductRequest *Library:* ClOdCommon *Class:* ClOdProductRequest	*Process:* EcPlOdMgr *Library:* PlOdMgrClient *Class:* PlOdMsgProxy	The EcPlOdMgr returns an order id to the CGI Interface to provide to the user to track the order.
Update Data	One per update/retrieve request	PDPS Database	*Process:* EcPlOdMgr *Libraries:* PlCore1, PlOdMgrClient *Class:* PlDataType	The EcPlOdMgr process sends requests to the **PDPS database** to update data defining a PGE. Requests are also sent for updates of granule information (location, size, etc.), processing status, and check pointing.
Submit Subscription	One per subscription created	*Process:* EcSbSubServer *Library:* EcSbSr *Class:* EcSbSubscription	*Processes:* EcPlOdMgr *Library*: PlCore1 *Class*: PlDataType	The EcPlOdMgr processes create subscriptions for an ESDT insert event and enter the subscriptions via the **EcSbSubServer**.

Table 4.6-4. PLANG CSCI Process Interface Events (14 of 19)

Event	Event Frequency	Interface	Initiated By	Event Description
Request Communications Support	One per process request.	***Process:*** EcCsIdNameServer ***Libraries:*** EcPf, Middleware, FoNs, FoIp, oodce *Classes:* EcPfManagedServer, EcPfClient, CCSMdwNameServer, FoNsNameServerProxy, CCSMdwRwNetProxy ***Library (Common):*** EcUr *Class:* EcUrServerUR ***Process:*** EcSbSubServer *Library:* EcSbSr *Class*: EcSbSubscription ***Library:*** EcDcMsgPsng1 ***Library:*** event *Class:* EcLgErrormsg ***Library:*** EcSeCmi Class: EcSeCmi ***Process:*** EcCsRegistry *Library:* EcCsRegistry *Class:* EcRgRegistryServer_C	*Process:* EcPlOdMgr *Library:* PlOdMgrClient PlOdMgrClientStub PlCore1 PlCore1IF PlCore1Sub PlCore2 PlUtil PlSSIT *Classes:* PlOdMgr.c, PlOdMsg.c, PlOdMgrClient.c	The **DCCI CSCI** provides a library of services available to each SDPS and CSMS CSCI. The CSCI services required to perform specific assignments are requested from the DCCI CSCI. These services include: • CCS Middleware Support • Database Connection Services • Network & Distributed File Services • Name/Address Services • Password Services • Server Request Framework (SRF) • Universal Reference (UR) • Error/Event Logging • Message Passing • Fault Handling Services • Mode Information • Query Registry - Retrieving the requested configuration attribute-value pairs from the Configuration Registry

Table 4.6-4. PLANG CSCI Process Interface Events (15 of 19)

Event	**Event Frequency**	**Interface**	**Initiated By**	**Event Description**
Export Location Information	Once at system startup and after each failure recovery	*Process:* EcCsIdNameServer *Libraries:* EcPf, Middleware, FoNs, FoIp, oodce *Classes:* EcPfManagedServer, EcPfClient, CCSMdwNameServer , FoNsNameServerPro xy, CCSMdwRwNetProxy	*Process:* EcPlOdMgr *Library:* PlOdMgrClient PlOdMgrClientStub PlCore1 PlCore1IF PlCore1Sub PlCore2 PlUtil PlSSIT *Classes:* PlOdMgr.c, PlOdMsg.c, PlOdMgrClient.c	The EcPlOdMgr stores physical and logical location information in the **EcCsIdNameServer**.
Password Seed	One per password seed	*Process:* Cryptographic Management Interface *Library:* EcSeCmi *Class:* EcSeCmi	*Process:* EcPlOdMgr *Libraries:* PlOdMgrClient, PlOdMgrClientStub, PlCore1, PlCore1IF, PlCore1Sub, PlCore2, PlUtil, PlSSIT *Classes:* PlOdMgr.c, PlOdMsg.c, PlOdMgrClient.c	The server provides a unique number as a seed to the Cryptographic Management Interface (within DCCI CSCI) for generating a password to the EcPlOdMgr processes.

Table 4.6-4. PLANG CSCI Process Interface Events (16 of 19)

Event	Event Frequency	Interface	Initiated By	Event Description
Filesystem Request	Once at system startup and after each failure recovery	*Process:* NFS Clients (COTS)	*Process:* EcPlOdMgr *Libraries:* PlOdMgrClient, PlOdMgrClientStub, PlCore1, PlCore1IF, PlCore1Sub, PlCore2, PlUtil, PlSSIT *Classes:* PlOdMgr.c, PlOdMsg.c, PlOdMgrClient.c	The NFS clients (via the EcPlOdMgr) request ECS files or directories via an established mount point. The NFS Server makes the storage device(s) and its data accessible for use by the clients.
Search Inventory	One per query	*Process:* EcDsScienceDataServer *Library:* DsCl *Class:* DsClQuery	*Process:* EcPlOdMgr *Library:* DpPrDssIF *Class:* DpPrDSSInterface	The EcPlOdMgr process creates two types of queries. One type only has the ESDT short name and data start and stop times and the other type also includes spatial coordinates. The EcPlOdMgr process creates an ESDT Reference from an UR after receiving a subscription notification or receiving an ESDT reference from a query. The EcPlOdMgr process queries when predicted data is not available. The **EcDsScienceDataServer** returns metadata information about the granule being inspected.

Table 4.6-4. PLANG CSCI Process Interface Events (17 of 19)

Event	**Event Frequency**	**Interface**	**Initiated By**	**Event Description**
Submit User Acquire	One per request	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClRequest, DsClCommand, DsClESDTReferenceCollector	*Process:* EcPlOdMgr *Library:* PlCore2 *Classes:* PlPrActivator, DpPrDSSInterface	The EcPlOdMgr submits an acquire command to the **EcDsScienceDataServer** on behalf of the user. The user gets a response via the EcDsDistributionServer upon data distribution.
Return URs	Per inventory query	*Process:* EcPlOdMgr *Library:* DpPrDssIF *Class:* DpPrDSSInterface	*Process:* EcDsScienceDataServer *Library:* DsCl *Class:* DsClQuery	The EcPlOdMgr receives Earth Science Data Type (ESDT) Universal References (URs) for the granules from the **EcDsScienceDataServer**.
Return Inventory Granule Metadata	All metadata per inventory search	*Process:* EcPlOdMgr *Library:* DpPrDssIF *Class:* DpPrDSSInterface	*Process:* EcDsScienceDataServer *Library:* DsCl *Class:* DsClQuery	The EcPlSubMgr receives the inventory granule metadata identifying the scene within the granule based on an inventory search request sent to the **EcDsScienceDataServer**.
Request On-Demand Tracking Elements	Per order	*Process:* EcMsAcOrderSrvr *Libraries:* MsAcClnt, MsAcComm	*Process:* EcPlOdMgr *Library:* PlCore2	The EcPlOdMgr requests on-demand tracking elements (i.e., Order ID and Request ID) from the **EcMsAcOrderSrvr**.
Request Standing Order Tracking Elements	Per order	*Process:* EcMsAcOrderSrvr *Libraries:* MsAcClnt, MsAcComm	*Process:* EcPlOdMgr *Library:* PlCore2	The EcPlOdMgr requests standing order tracking elements (i.e., Order ID and Request ID) from the **EcMsAcOrderSrvr**.
Return Order ID	Per order	*Process:* EcPlOdMgr *Library:* PlCore2	*Process:* EcMsAcOrderSrvr *Libraries:* MsAcClnt, MsAcComm	The **EcMsAcOrderSrvr** sends an order id to the EcPlOdMgr to allow the user to track on-demand product or order requests.

Table 4.6-4. PLANG CSCI Process Interface Events (18 of 19)

Event	Event Frequency	Interface	Initiated By	Event Description
Return Configuration Parameters	One set per request	*Process:* EcPlOdMgr *Libraries:* PlOdMgrClient, PlOdMgrClientStub, PlCore1, PlCore1IF, PlCore1Sub, PlCore2, PlUtil, PlSSIT *Classes:* PlOdMgr.c, PlOdMsg.c, PlOdMgrClient.c	*Process:* EcCsRegistry *Library:* EcCsRegistry *Class:* EcRgRegistryServer_C	The **EcCsRegistry** returns the attribute-value pairs (configuration parameters) to the EcPlOdMgr upon request.
Import Location Information	As required for processing	*Process:* EcPlOdMgr *Libraries:* PlOdMgrClient, PlOdMgrClientStub, PlCore1, PlCore1IF, PlCore1Sub, PlCore2, PlUtil, PlSSIT *Classes:* PlOdMgr.c, PlOdMsg.c, PlOdMgrClient.c	*Process:* EcCsIdNameServer *Libraries:* EcPf, Middleware, FoNs, FoIp, oodce *Classes:* EcPfManagedServer, EcPfClient, CCSMdwNameServer, FoNsNameServerProxy, CCSMdwRwNetProxy	The EcPlOdMgr requests server location information from the **EcCsIdNameServer**.

Table 4.6-4. PLANG CSCI Process Interface Events (19 of 19)

Event	Event Frequency	Interface	Initiated By	Event Description
Server Account and random password	One per account and password	*Process:* EcPlOdMgr *Libraries:* PlOdMgrClient, PlOdMgrClientStub, PlCore1, PlCore1IF, PlCore1Sub, PlCore2, PlUtil, PlSSIT *Classes:* PlOdMgr.c, PlOdMsg.c, PlOdMgrClient.c	*Process:* Cryptographic Management Interface *Library:* EcSeCmi *Class:* EcSeCmi	The Cryptographic Management Interface generates a random password for the account based on the seed provided by the EcPlOdMgr process.
Request Management Services	At system startup or shutdown and for restarts	*Process:* EcPlOdMgr	DAAC unique startup scripts	**System startup and shutdown -** Please refer to the release-related, current version of the Mission Operations Procedures for the ECS Project document (611) and the current ECS Project Training Material document (625), identified in Section 2.2.1 of this document.

4.6.1.6 Production Planning Data Stores

Table 4.6-5 provides descriptions of the production planning data stores shown in the PLANG CSCI architecture diagram.

Table 4.6-5. PLANG CSCI Data Stores

Data Store	Type	Functionality
PDPS Database	Database	The PDPS database is replicated within each site for fault handling purposes. This PDPS database holds all the persistent data (and facilitates the sharing of this data) including, but not limited to: • Resource information entered with the Resource Planning utilities • PGE and data type information entered at SSIT • Production Request, Data Processing Request and Data Granule Information entered using the Production Request Editor • Plan information entered using the Production Planning Workbench • Task recovery information • Production strategies entered using the Production Strategy GUI The PDPS database also provides security, fault tolerance, and verifies requests for concurrent access to data.

4.6.2 Planning Subsystem Hardware Components

4.6.2.1 Planning Hardware CI (PLNHW) Description

The PLNHW hardware (PLNHW) consists of a server with the Sybase database management system (DBMS) and the workstations to support the Operations staff by providing the Planning Workbench.

The PDPS DBMS Server runs on either a four processor SUN Server or a dual-processor Sun workstation (see 920-TDx-001 series of baseline documents) with 64-bit Ultra-SPARC processors. Each PDPS DBMS Server is equipped with 512 MB of memory (see 920-TDx-001 series of baseline documents) required by the workstation processors and the Sybase DBMS.

The internal disks provide swap space and file system space for the operating system and the file space for applications software (see 920-TDx-001 series of baseline documents).

Either a Storage Array or an appropriately sized storage unit configured for the database application provides storage for the PDPS database. These storage units are attached to the host via a fast-wide small computer system (SCSI). Additionally, this storage unit backs up the Queuing Server database (see Section 4.7.3, Data Processing Hardware Components).

4.6.2.2 Planning Workstation Description

The Planning workstation contains and runs the Planning Workbench software. The Planning Workbench is the Production Planning function and the Resource Planning function. One or more workstations are used to run Production Planning and/or Resource Planning at each site based on the site size.

The Planning workstation is a SUN workstation class machine with either a single SPARC or Ultra-SPARC based processor (see 920-TDx-001 series of baseline documents).

The Planning workstation has 384 MB of memory (see 920-TDx-001 series of baseline documents) and each has a fast-wide SCSI controller to attach to a storage subsystem.

The internal disks provide swap space for the operating system and file system space for the operating system and applications (see 920-TDx-001 series of baseline documents). Additional storage for the Planning workstations can be attached to the SCSI controller.

Sites generally have a minimum of two Planning workstations. If a planning workstation fails, another planning workstation assumes the Planning Workbench functionality from the failed workstation. In cases with one Planning workstation, the Planning workstation tasks are assumed by an available equivalent workstation. Faulty hardware is either repaired or replaced by a certified technician.

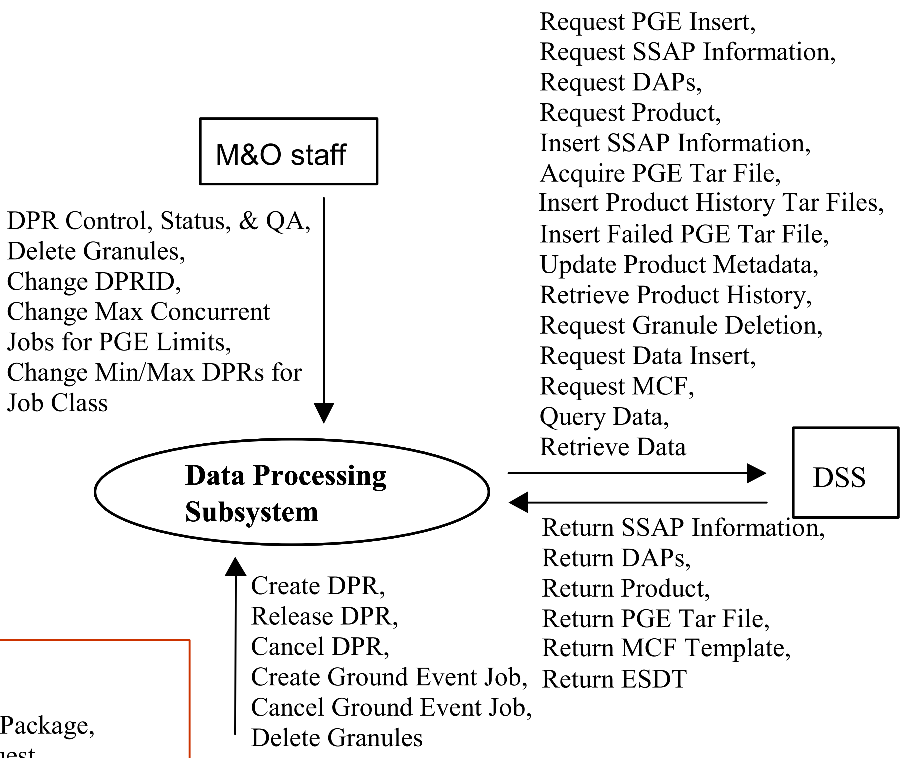
4.7 Data Processing Subsystem Overview

The Data Processing Subsystem (DPS) provides the Data Processing capabilities at each ECS site. The DPS capabilities include:

- A queued processing environment to support data product generation. The DPS executes DPRs on available processing resources, as an associated processing job containing all the information needed to accomplish the processing. DPRs are submitted by the PLS and triggered by the arrival of data or triggered internally by the PLS (i.e., reprocessing). PGEs resulting from the integration and test of delivered science algorithms [ref.: ECS White Paper 193-WP-118] and encapsulated into the SDPS with the Science Data Processing (SDP) Toolkit are used by DPRs to process data. User-specified methods are also used for processing specific data types.
- The Operational interfaces required to monitor the execution of the science software (PGEs).
- Support for science algorithm execution via the SDP Toolkit. The SDP Toolkit is a set of tools to provide a common interface for encapsulating each science algorithm into the SDPS environment. (See the SDP Toolkit Users Guide for the ECS Project (333-CD-605-003)) and PGS Toolkit Requirements Specification for the ECS Project (193-801-SD4-001, a.k.a. GSFC 423-16-02) for guidance on the roles and responsibilities of the SDP Toolkit to support the execution of science software.
- Support for the preliminary format processing of data sets (L0 data products) required by the science algorithms.
- Providing an Algorithm Integration and Test (AI&T) environment to integrate new science algorithms, new versions of existing science algorithms, and user methods into the SDPS environment. The system acquires the algorithm or method via an ingest process reflecting local site policies for acceptance of software for integration into the environment. (See Section 4.7.2 “Algorithm Integration and Test Tools (AITTL) CSCI Description).
- The DAAC Quality Assurance (QA) procedures and conditions to verify each data product by the scientific personnel at each DAAC. All data products, both those generated by and input to a submitted job, are available for examination by DAAC scientific personnel to verify data content to be in accordance with quality standards set by the DAAC.

Data Processing Subsystem Context

Figure 4.7-1 is the Data Processing Subsystem context diagrams. The diagrams show the events sent to the DPS and the events the DPS sends to other SDPS and CSMS subsystems and the Operations staff.



Note:
DAP =Delivered Algorithm Package,
DPR =Data Processing Request,
ESDT =Earth Science Data Type,
MCF =Metadata Configuration File,
PGE =Product Generation Executable,
QA =Quality Assurance (validation),
SSAP =Science Software Archive Package

Figure 4.7-1. Data Processing Subsystem Context Diagram

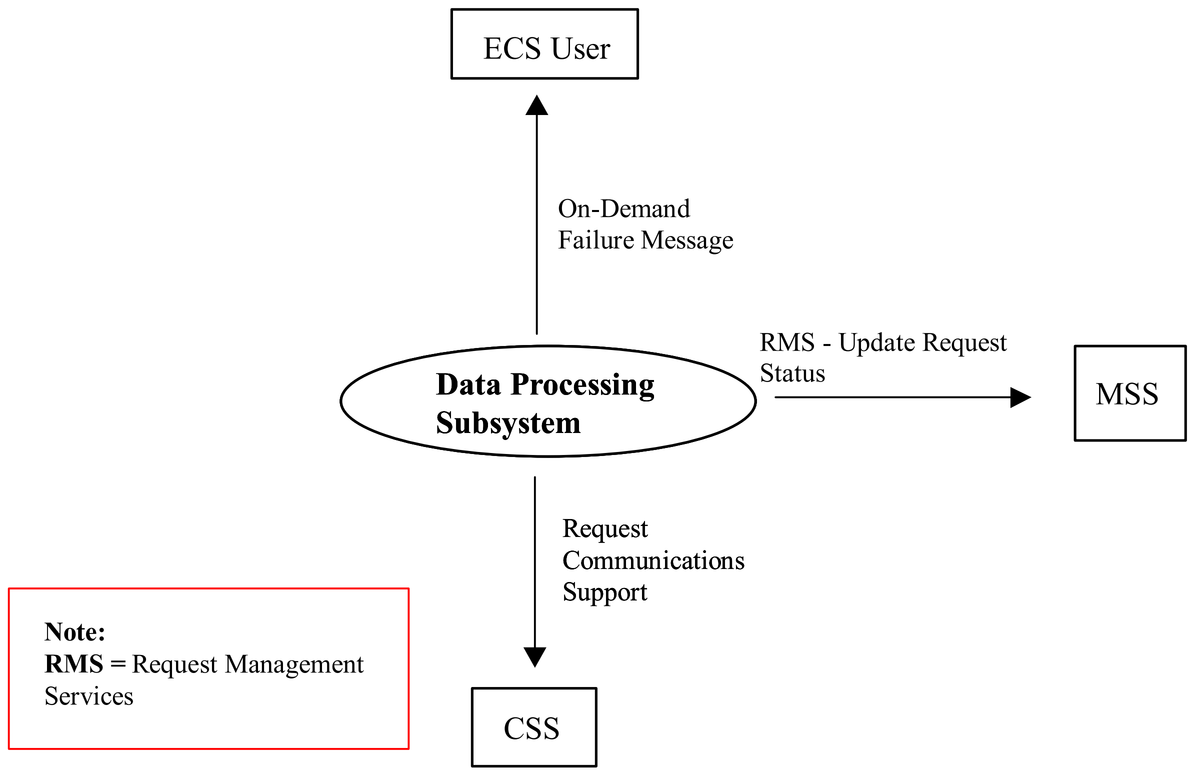

ECS User
On-Demand Failure Message
Data Processing Subsystem
RMS - Update Request Status
MSS
Request Communications Support
CSS
Note:
RMS = Request Management Services

Figure 4.7-1. Data Processing Subsystem Context Diagram (cont.)

Table 4.7-1 provides descriptions of the interface events shown in the Data Processing Subsystem context diagrams.

Table 4.7-1. Data Processing Subsystem Interface Events (1 of 4)

Event	Interface Event Description
DPR Control, Status, & QA	The **M&O staff** provides Data processing control and supports Data Processing Request (DPR) status and Quality Assurance activities.
Delete Granules	The **M&O staff** sends requests to the DPS to delete granules associated with cancelled DPRs.
Change DPR ID	The **M&O staff** can change the DPR ID of an existing DPR.
Change Max Concurrent Jobs for PGE Limits	The **M&O staff** can add to or update values in the DpPrPgeLimits database table.
Change Min/Max DPRs for Job Class	The **M&O staff** can add to or update values in the DpPrClassSchedulingLimits database table.

Table 4.7-1. Data Processing Subsystem Interface Events (2 of 4)

Event	Interface Event Description
Request PGE Insert	The DPS sends requests to the **DSS** to insert data that defines a Product Generation Executable (PGE) and allows it to be scheduled and executed.
Request SSAP Information	The DPS sends requests to the **DSS** for Science Software Archive Package (SSAP) information, including names of existing SSAPs and the information associated with a specific SSAP. In response, the DSS sends lists of SSAPs and related information.
Request DAPs	The DPS requests DAPs based on Universal References (URs) from the **DSS**. The DAPs are placed on a local DPS disk.
Request Product	The DPS sends requests, to the **DSS**, for particular data granules to be pushed, via the File Transfer Protocol (FTP) service, onto the DPS science processor as input for data processing or for Science System Integration and Test (SSIT) work.
Insert SSAP Information	The Operations staff sends requests to the **DSS** to insert SSAP information, via the DPS SSAP Graphical User Interface (GUI), including SSAP name, SSAP version number, PGE name, PGE version number, and SSAP Acceptance Date.
Acquire PGE tar file	The DPS acquires a tar file for any PGE not currently local to the science processor from the **DSS**. The executable is extracted from the tar file and used during PGE execution.
Insert Product History Tar Files	The DPS sends a request to the **DSS** to insert the PGE Production History Tar File resulting outputs for permanent archive after the PGE has successfully completed executing.
Insert Failed PGE Tar File	After an unsuccessful execution of a PGE, the DPS obtains the Tar file containing the PGE log files, core dump (if any), Process Control File (PCF) and other files, and requests the files be inserted into the **DSS** for permanent archive.
Update Product metadata	The Operations Staff uses the Quality Assurance (QA) Monitor GUI in the DPS to send requests to update product metadata in the DSS.
Retrieve Product History	The Operations Staff uses the QA Monitor GUI to submit requests to the **DSS** to transfer the Production History tar file from the Science Data archives to the user's host machine.
Request Granule Deletion	The DPS sends delete requests, to the **DSS**, for particular granules (interim data) in the archive and the associated metadata to be deleted from the SDSRV inventory.
Request Data Insert	The DPS sends requests to the **DSS** to insert a particular file or files into the archive, and catalog the associated metadata in the SDSRV inventory. These files can be processing output, static files received with PGEs, PGE tar files, Algorithm Packages (APS), SSAPs or Delivered Archive Packages (DAPs), failed PGE tar files, or production history files.
Request MCF	The INS and DPS request the Metadata Configuration File (MCF) template, from the **DSS**, prior to a data insert request.
Query Data	The DPS submits requests of this type to the **DSS**. It searches the archive for granules that match the user-supplied selection criteria: data type and begin/end date. Results are displayed to the user.
Retrieve Data	The DPS sends retrieval requests, to the **DSS**, for a particular data granuleId. The product is transferred (pushed), via the FTP service, onto the DPS science processor and used as input for PGE processing or for SSIT work.

Table 4.7-1. Data Processing Subsystem Interface Events (3 of 4)

Event	Interface Event Description
Return SSAP Information	The DPS receives lists of SSAPs and related information from the **DSS**.
Return DAPs	The DAPs are placed on a local DPS disk by the **DSS**.
Return Product	The data granules requested by the DPS are sent from the **DSS**.
Return PGE Tar File	After an unsuccessful execution of a PGE, the DPS obtains the Tar file containing the PGE log files, core dump (if any), Process Control File (PCF) and other files, and requests the files be inserted into the **DSS** for permanent archive.
Return MCF Template	The **DSS** provides the MCF template, to the DPS, to populate as part of the GetMCF service call.
Return ESDT	The **DSS** returns the requested ESDT to the DPS.
Create DPR	The DPS uses the dprId from the **PLS** to insert a job box for a DPR into AutoSys if all the required input data is available. If the input data is not available, information used to construct the job in AutoSys is queued by the Job Management CSC until a release DPR is received.
Release DPR	The DPS uses the dprId from the **PLS** to release jobs currently waiting for external data in the Job Management queue into AutoSys.
Cancel DPR	The DPS uses the dprId from the **PLS** to delete jobs in AutoSys or from the queue.
Create Ground Event Job	The DPS uses the ground event Id from the **PLS** to create a ground event job in AutoSys.
Cancel Ground Event Job	The DPS uses the ground event Id from the **PLS** to delete a ground event job in AutoSys.
Delete Granules	The **PLS** sends requests to the DPS to delete granules associated with cancelled DPRs.
On-Demand Failure Message	The DPS informs the **ECS user** (the submitter) of an On-Demand Processing Request when such a request has an unrecoverable failure.
Request Management Services	The **MSS** provides a basic management library of services to the subsystems, implemented as client or server applications, using the CSS Process Framework. The basic management library of services includes: • **System startup and shutdown** - Please refer to the release-related, current version of the Mission Operations Procedures for the ECS Project document (611) and the current ECS Project Training Material document (625), identified in Section 2.2.1 of this document. • **Update Request Status** - The DPS informs the **MSS** to update the status of an On-Demand Processing Request, when such request changes status (i.e., from Running to Completed or from Running to Failure).

Table 4.7-1. Data Processing Subsystem Interface Events (4 of 4)

Event	Interface Event Description
Request Communications Support	The **CSS** provides a library of services available to each SDPS and CSMS subsystem. The subsystem services required to perform specific assignments are requested from the CSS. These services include: • CCS Middleware Support • Database Connection Services • Network & Distributed File Services • Name/Address Services • Password Services • Server Request Framework (SRF) • Universal Reference (UR) • Error/Event Logging • Fault Handling Services • Mode Information • Query Registry - Retrieving the requested configuration attribute-value pairs from the Configuration Registry

The DPS has an internal interface to the COTS software product AutoSys. The DPS creates, starts, and deletes job boxes in AutoSys via this interface.

The PLS determines the processing activities required to generate data products specified by the Operations staff in a Production Request. Each processing activity is called a Data Processing Request (DPR). The PLS creates, releases or cancels DPRs in AutoSys via the DPS.

The DSS accesses the data archives via authorized user requests. The DPS requests the required input data for a Product Generation Executable (PGE) and Metadata Configuration Files (MCFs) from the DSS. The DPS also inserts PGE generated products, provides product production histories, and provides failed PGE information for debugging purposes. The DPS uses the DSS as a permanent repository for PGE tar files, Algorithm Packages (APs), Science Software Archive Packages (SSAPs) and Delivered Algorithm Packages (DAPs).

Data Processing Subsystem Structure

The DPS is comprised of three CSCIs:

- The Processing (PRONG) CSCI manages and monitors the Science Data Processing (SDP) environment to execute Science Software and algorithms (called PGEs) and generates data products.
- The Algorithm Integration and Test Tools (AITTL) CSCI is a set of tools for test and integration of new science software, new versions of existing science software, and user methods in the SDP operational environment. AITTL combines custom developed code with COTS software starting from a central application called the SSIT Manager.
- The SDP Toolkit (SDPTK) CSCI provides a set of software libraries to integrate Science Software into the SDPS environment. By promoting the POSIX standard, these libraries allow the SDP environment to support the generation of data products in a heterogeneous

computer hardware environment. (See SDP Toolkit Design Specification (455-TP-001-001) for the SDPTK architecture).

Use of COTS in the Data Processing Subsystem

- **Computer Associate's AutoSys** is a job scheduling software application to automate operations in a distributed UNIX environment. AutoSys performs automated job control functions for scheduling, monitoring, and reporting on the jobs residing on any Unix machine attached to an ECS network on the Science Data Processing hardware. AutoSys provides job-scheduling support with an Operator Console for monitoring and human intervention in the job stream. The Operator console allows the M&O staff to restart failed jobs and to view the status of events related to the job's execution. The Operator console includes an alarm manager, set in the job definition, to assist the Operations staff when responding to fault situations.

- **Computer Associate's AutoXpert** is a GUI providing different methods of viewing a job schedule progress. Noting color changes on the JOBSCAPE GUI can monitor the progression of DPR execution. Failed jobs can be detected and restarted if the job has failed due to the unavailability of an external resource. The HostScape GUI can be used to view the status of the science processors.

- **Sybase Adaptive Server Enterprise (ASE)**

  The Sybase ASE provides the capabilities to insert, update and delete PDPS database content. The Sybase ASE must be operational during the DPS operations.

- **CCS Middleware Client**

  CCS Middleware Client provides DSS with communications between other subsystems. CCS Middleware can reside on one or both sides of the interface. An instance must be installed on the platform where DSS resides. Although the CCS Middleware Client is part of CSS, this COTS product must be installed for DSS to run in the SDPS operational and test environment.

The DPS provides the hardware resources for science software execution, queuing, dispatching, and managing in a distributed environment of computing platforms. The DPS hardware consists of three CIs:

- Science Processor - The Science Processor HWCI (SPRHW) contains processing resources (central processing units, memory, disk storage, and input/output subsystems) necessary to perform first-time processing, reprocessing, and Algorithm Integration and Test (AI&T). Also, SPRHW provides the hardware resources (a Queuing Server) to support management of the science processes.

- Algorithm Quality Assurance - The Algorithm Quality Assurance HWCI (AQAHW) supports the DAAC Operations staff in performing the planned science and non-science product data quality validation procedures.

- Algorithm Integration and Test - The AI&T HWCI (AITHW) resources provide the operating system and support for the integration and test of science software at each DAAC. AITHW is the workstations and hardware tools required for software integration and test. AITHW does not, in this case, provide the computer capacity required for science software test (SPRHW provides the test capacity).

Error Handling and processing

EcUtStatus is a class used throughout the ECS custom code for general error reporting.

The Data Processing Subsystem (DPS) uses the EcUtStatus class for general error handling. It is almost always used as a return value for functions and allows detailed error codes to be passed back up function stacks.

The PRONG and AITTL CSCIs use two main mechanisms for error handling.

1. Return Values

   Functions can return an EcUtStatus object, which can be used to indicate a general success/failure status. Also, more detailed information on the exact reason for the failure can be provided. This is the most widely used mechanism within the PRONG and AITTL and in general these errors get propagated back up to the top-level functions with ALOG error messages being generated along the way.

2. Exceptions

   Some functions (for example, class constructors) cannot return values to indicate success or failure. These functions can throw exceptions. These errors are usually caught by other functions at a low level and converted into EcUtStatus return values (as described in 1).

Currently, the PRONG and AITTL CSCIs client interfaces only support returning error messages back to client programs, along with a generic success/failed status.

In addition, the PRONG and AITTL use the following method of tracking errors; stored procedure tracing. Some PRONG and AITTL stored procedures write error and informational messages to a table in the Planning and Data Processing Subsystems (PDPS) database named DpPrTrace. The stored procedure ProcInsertTrace is used to do this. A trigger on this table truncates the messages after 10,000 messages have been written to the table. There are no other specialized error libraries and classes used by the PRONG and AITTL.

Column	Example
Time	Apr 2 2002 14:01:27.960000
Procedure Name	ProcGetReadyDPRs
Called From	(NULL)
Dpr Id	MODPGE02#s28021500OPS
Message	99 pgeId slots available for pgeId MODPGE02#syn#001

Example Row from the DPS DpPrTrace Table

For writing messages to the Applications Log (ALOG), the following functions are used:

EcLgLogError sends a message to the ALOG at severity level 1. For example, EcLgLogError(methodName, returnStatus.GetLogMessageLink(),

    "Error: unable to copy tar file from '%s%s' to '%s' ",

    dataserverpath.data(), pgeTarName.data(), destinationPath.data());

EcLgLogWarning sends a message to the ALOG at severity level 2. For example, EcLgLogWarning(methodName, lstatus.GetLogMessageLink(), "error terminating DPR job") ;

EcLgLogInformational sends a message to the ALOG at severity level 3. For example, EcLgLogInformational(methodName,

    status.GetLogMessageLink(),"not using DpPrAutoSysProfile");

For writing messages to the debug log, the following macros are used:

PF_STATUS writes a message at a "log level" of 1 to the debug log. For example,

PF_STATUS { cerr << methodName << "Can't retrieve DpPrDeleteFailedPGEJobs";

cerr << " param. -- setting to FALSE" << endl;}

PF_VERBOSE writes a message at a "log level" of 2 to the debug log. For example,

PF_VERBOSE {cerr << methodName <<"urName = " << urName << endl;}

PF_DEBUG writes a message at a "log level" of 3 to the debug log. For example,

PF_DEBUG {cerr << methodName << "can not get PF config file pointer!" << endl;}

TOOLKIT Error/Status Reporting (SMF Tools)

To detect and report error and status conditions in a consistent manner, standardized status messages and status codes must first be established. The method used to institute these message/code pairs is by way of the ‘smfcompile’ utility. But first, users need to create Status Message Files (SMFs) to contain their custom status messages and corresponding status identifiers. These identifiers take the form of user defined mnemonics that visually convey the essence of the status message. The user makes direct use of these mnemonics in their software when testing for status conditions and when interfacing with the SMF Toolkit functions. Once an SMF is completed, the smfcompile utility is run to bind the status messages and mnemonics with integral status codes. This process facilitates the run-time access of all status messages and provides for the referencing of status mnemonics within the user's code.

The status codes generated by the 'smfcompile' utility are guaranteed to be unique to ensure that there are no ambiguous status conditions, in the event that code fro different Science Computing Facilities (SCFs) is merged into a single executable and/or PGE. This uniqueness is possible because "seed" values, which are different for every SMF, are used in the generation of the status codes. Typically, many SMF files are created in the course of software development; therefore many seed numbers are required. However, it is important to note that valid seed numbers can only be obtained from the Toolkit development team. Any attempt to produce SMFs from "home-grown" seed values can result in the SMFs being unusable at integration & test time.

The SDP Toolkit routines actually contain their own collection of status codes and associated status messages for describing the state of each Toolkit function. Users of the Toolkit functions should examine the return values of each tool before performing any other action. To inform a calling unit (user's software) about the exit state of a called Toolkit routine, each Toolkit function sets a status message and assigns a status code to the return value. On the basis of its interpretation of this return value, the calling unit may elect to perform some error handling. As part of this procedure, the user should either propagate the existing status code up through their calling hierarchy, or set a status code and message to represent the outcome of any local error handling attempt.

Upon detection of an error state, users are advised to report on the existing error prior to performing an error handling procedure. The content of these reports might include the following:

- A user-defined message string to convey the nature of the status condition
- A user-defined action string to indicate the next operation to be performed in response to the status condition
- A system defined string that uniquely identifies the environment in which the status condition occurred.

However, this is merely a suggestion; the users are free to define the content of the status reports to satisfy their own requirements. The method for reporting this information involves the generation of a report from the information just described and the subsequent transmission of that report to the appropriate destination(s).

The tools provided here allow for the propagation of status information within a PGE executable to facilitate a user's error handling process. They also provide the means to communicate status and error information to various monitoring authorities and event logs. Additionally, there is a tool that enables the user to specify, a priori, the action to be taken in the wake of a fatal arithmetic event. This mechanism allows users to take their own corrective measures to control an event that is terminal by default. Note that all other event conditions fall under the purview of system processing and are thereby controlled by the governing SDPS software.

Several new features have been incorporated into these tools for Toolkit 5 in order to improve their efficiency. One of those features allows for the buffering of individual status messages up to some user defined run-time limit. This should greatly reduce the amount of I/O required to access these messages. As a process proceeds to completion, new status messages are buffered as older, less used status messages become un-buffered. The goal here is to only access status messages from their run-time file when they are being referenced for the first time. The actual observed improvement depends on the degree to which a process' status messages are localized (i.e., A particular status message should ideally only be referenced within a short body of code.) and the buffer size, which is set by the user. Another feature reduces the number of replicated status messages that can appear in the status log file. This is accomplished by "compressing" duplicate messages into a count of such messages. This feature should significantly reduce the size of the status log file and contribute to its general readability.

Since each function has only one return value, every effort has been made to preserve the most important warning or error value on returning. Given that subordinate functions often have several possible returns, and different users have different priorities, it is always advisable to check the message log as well as examining the return. When totally inconsistent behavior is found in a return from a subordinate function, the returned value is PGS_E_TOOLKIT. Example: a Toolkit function passes an internally generated vector, whose length is certain to be nonzero, to a subordinate function. The lower-level function then returns a warning or error return stating that the vector is of zero length; while the higher-level function returns PGS_E_TOOLKIT. Another example: if a valid spacecraft tag is passed in, but rejected as invalid down the processing line, the error PGS_E_TOOLKIT is returned by the higher-level function. Thus return value PGS_E_TOOLKIT indicates a flaw in the software, the violation of an array boundary, hardware, compiler, or system error, corrupted data, or some similarly serious condition that invalidates the processing.

Logging Control

PCF entry:
10114|Logging Control; 0=disable logging, 1=enable logging|1

This can be used to disable logging altogether. If logging is disabled NO message will output to any log files (although a small header is still written to the log files indicating logging for this PGE has been disabled). The Default State is for logging to be enabled.

Trace Control

PCF entry:
10115|Trace Control; 0=no trace, 1=error trace, 2=full trace|0

This can be used to specify the trace level for message logging. Tracing is a feature made possible by the addition of two SMF tools: PGS_SMF_Begin and PGS_SMF_End. Users may include these tools at the beginning and ending of their functions (respectively) to signal to the

SMF system when each user defined function is entered and exited. Three levels of tracing are possible:

No Tracing

This is the Default State. No information concerning the entering or exiting of functions is recorded to the log files. No information concerning the path of a function call is recorded to the log files.

Example Log Entry:

```
func4():PGSTD_W_PRED_LEAPS:27652
predicted value of TAI-UTC used (actual value unavailable)
```

Error Tracing

If error tracing is enabled, information concerning the path of a function call is recorded to the log files any time a status message is logged to a log file. This is useful in determining where in a chain of function calls an error occurred. No information concerning the entering or exiting of functions is recorded in this state.

Example Log Entry:

```
main():
   func1():
      func2():
         func3():
            func4():PGSTD_W_PRED_LEAPS:27652
            predicted value of TAI-UTC used (actual value unavailable)
```

Full Tracing

If full tracing is enabled, a message is written to the log files each time a function is entered and exited (only those user functions with the PGS_SMF_Begin/End calls, see above). Indenting is also done to show the path of each function call.

Example Log Entry:

```
PGS_SMF_Begin: main()

  PGS_SMF_Begin: func1()

    PGS_SMF_Begin: func2()

      PGS_SMF_Begin: func3()

        PGS_SMF_Begin: func4()

          func4():PGSTD_W_PRED_LEAPS:27652
          predicted value of TAI-UTC used (actual value unavailable)

        PGS_SMF_End: func4()

PGS_SMF_End: func3()
```

PGS_SMF_End: func2()

PGS_SMF_End: func1()

PGS_SMF_End: main()

Process ID Logging

PCF entry:
10116|Process ID logging; 0=don't log PID, 1=log PID|0

This can be used to enable the tagging of log file entries with the process ID of the process from which the entry came. This is useful for PGEs that run concurrent processes, which are all writing to a single log file simultaneously. If process ID logging is enabled, each log entry is tagged with the process ID of the process making the entry. This can facilitate in post-processing a log file.

Example Log Entry:
func4():PGSTD_W_PRED_LEAPS:27652 (PID=2710)
predicted value of TAI-UTC used (actual value unavailable)

Status Level Control

PCF entry:
10117|Disabled status level list (e.g., W S F)|<status level list>

This can be used to disable the logging of status codes of specific severity levels. A list of levels to be disabled should be substituted for <status level list> (e.g.: N M U). No message of a status level indicated in the list is recorded to any log. The Default State is to enable logging for all status levels.

Status Message Seed Control

PCF entry:
10118|Disabled seed list|<status code seed list>

This can be used to disable the logging of status codes generated from specific seed values. A list of seed values, the status codes derived from which should be disabled, should be substituted for <status code seed list> (e.g.: 3 5). No message derived from a seed value indicated in the list is recorded to any log. The Default State is to enable logging for all seed values.

Individual Status Code Control

PCF entry:
10119|Disabled status code list|<status code list>

This can be used to disable the logging of specific status codes. A list of status code mnemonics and/or numeric status codes should be substituted for <status code list> (e.g.: PGSTD_M_ASCII_TIME_FMT_B 678954). Note that using mnemonics can disable only

Toolkit status codes. To disable a user generated status code a numeric status code must be used. No messages whose status codes or mnemonics included in the list are recorded to any log file. The Default State is to enable logging for all status codes.

Generating Run-time E-Mail Messages

A PGE may be configured to automatically generate and send e-mail messages during run-time when specific user defined status codes are logged. This is done by assigning an e-mail action to a given user defined status code.

An e-mail action is an SMF code with the special status level of "C" and a mnemonic that begins with the characters "PGSEMAIL" (the rest of the mnemonic may contain any other valid mnemonic characters), for example:

PGS_*C_PGSEMAIL*_SEND_EMAIL
ASTER_*C_PGSEMAIL*_ALERT
MODIS_*C_PGSEMAIL*_ERROR

An e-mail message is generated anytime a user defined status code with an associated e-mail action is logged via the SMF logging routines. The contents (body) of these messages are the text (message) associated with the user defined status code. The subject of these messages is the mnemonic associated with the user defined status code. The list of recipients is defined in the e-mail action definition.

Example:
In a user defined status message file the following status code mnemonic label and e-mail action mnemonic label have been defined (the e-mail action is associated with the status code via the "::" syntax):

```
MODIS_E_PGE_INIT_FAILED   The PGE failed to initialize.
                          ::MODIS_C_PGSEMAIL_NOTIFY
MODIS_C_PGSEMAIL_NOTIFY   john@modis.org, sue@modis.org
```

The following lines appear in a C source code file:

```
    returnStatus = initializePGE();
    if (returnStatus == MODIS_E_PGE_INIT_FAILED)
    {PGS_SMF_SetStaticMsg(returnStatus, "main()");
        exit(1);}
```

At run-time, if the returned status code from the function initializePGE() has the value defined by MODIS_E_PGE_INIT_FAILED, this status is logged via the SMF function PGS_SMF_SetStaticMsg(), and because this status code has an e-mail action associated with it, an e-mail message is generated.

The e-mail message is sent to: sue@modis.org and john@modis.org
the subject field of the e-mail message is : MODIS_E_PGE_INIT_FAILED
The text of the e-mail message is : The PGE failed to initialize.

Note:
This functionality will be disabled at the DAACs.

4.7.1 Processing Software Description

4.7.1.1 Processing Functional Overview

The Processing (PRONG) CSCI initiates, monitors, and manages the execution of science software algorithms (referred to as PGEs). The PRONG CSCI is informed of the required execution of a PGE through a DPR received from the PLS. When all necessary input data becomes available, PRONG initiates the execution of the PGE. (N.B.: Some or all input data can reside in a Data Server not at the DAAC site.)

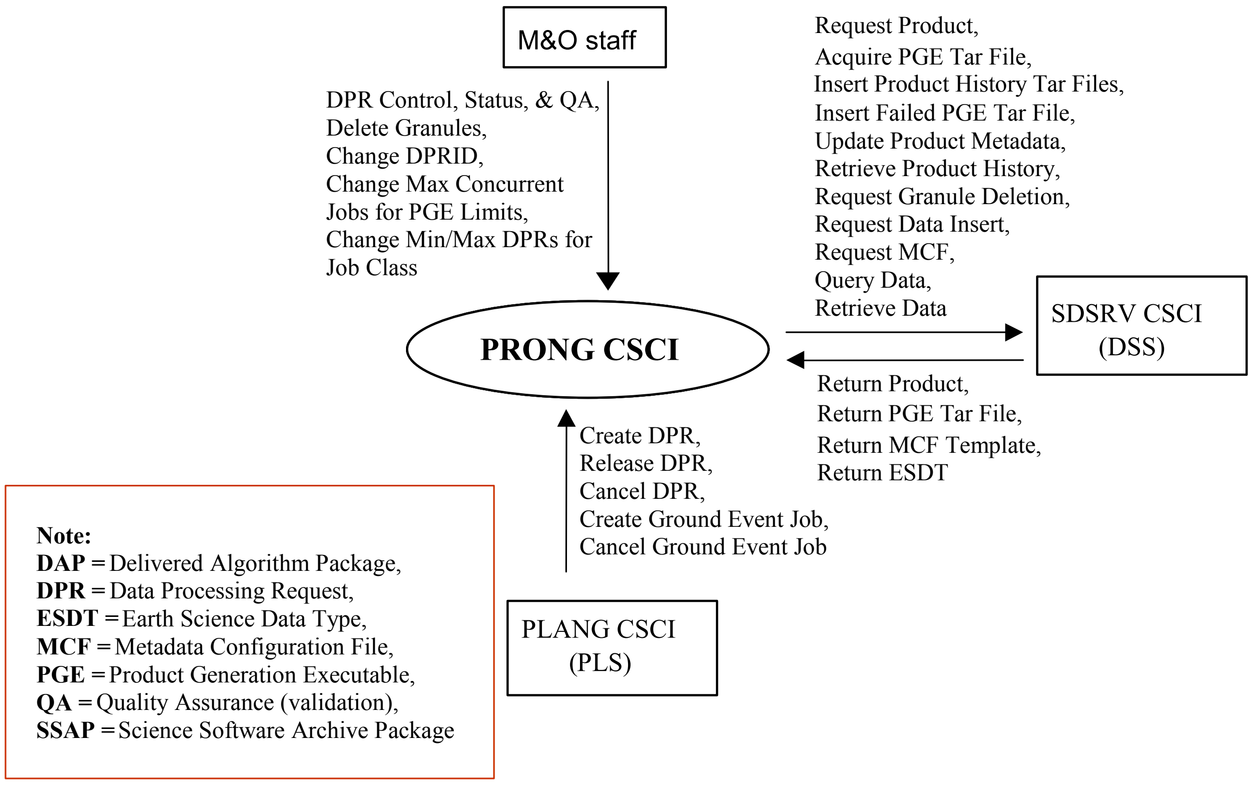
The PRONG CSCI has the following capabilities:

- Manages execution of science software algorithms
- Manages SDP computer hardware resources
- Manages the data flow required to execute a science software algorithm
- Manages the data flow generated by the execution of a science software algorithm
- Monitors processing status, and allows manual intervention, when necessary, in the SDP operations environment, including processing queue control
- Supports validation of product data quality
- Provides status and user updates (in the event of a failure) for On-Demand Processing Requests

4.7.1.2 Processing Context

Figure 4.7-2 is the PRONG CSCI context diagrams. The diagrams show the events sent to the PRONG CSCI and the events the PRONG CSCI sends to other CSCIs and the Maintenance and Operations (M&O) staff.

Table 4.7-3 provides descriptions of the interface events shown in the PRONG CSCI context diagrams.



Note:
DAP = Delivered Algorithm Package,
DPR = Data Processing Request,
ESDT = Earth Science Data Type,
MCF = Metadata Configuration File,
PGE = Product Generation Executable,
QA = Quality Assurance (validation),
SSAP = Science Software Archive Package

Figure 4.7-2. PRONG CSCI Context Diagram

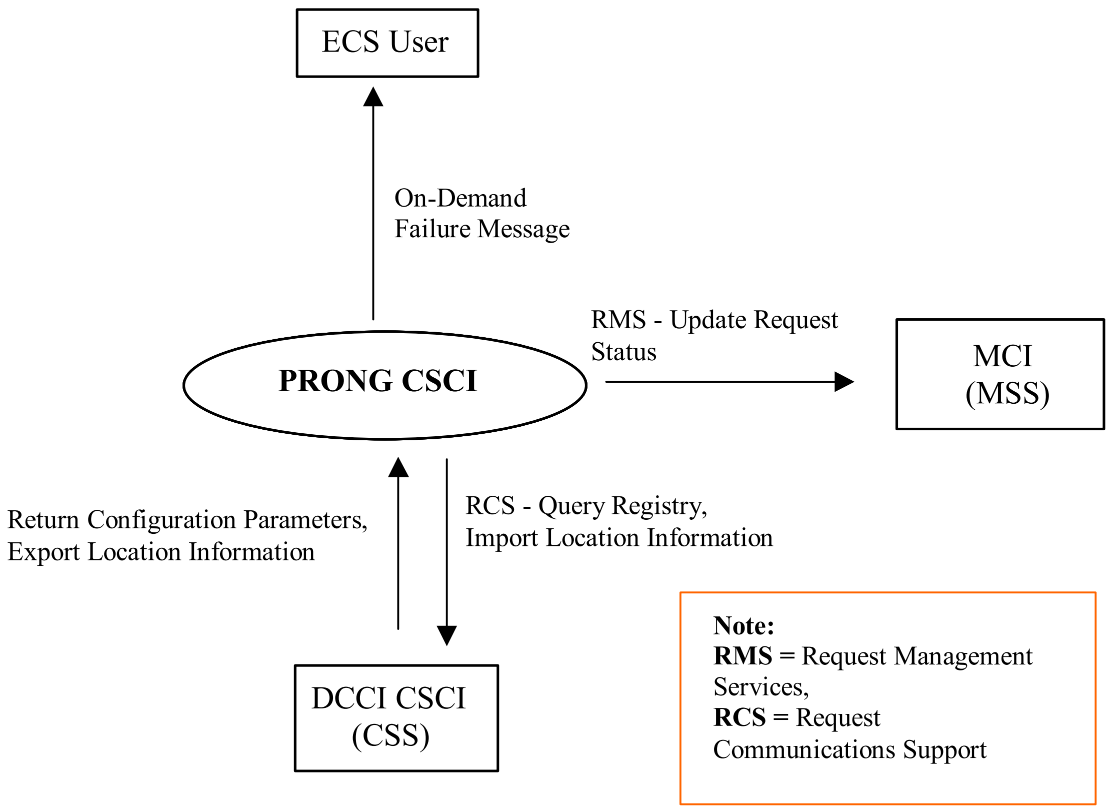

ECS User
On-Demand Failure Message
RMS - Update Request Status
PRONG CSCI
MCI (MSS)
Return Configuration Parameters, Export Location Information
RCS - Query Registry, Import Location Information
DCCI CSCI (CSS)
Note:
RMS = Request Management Services,
RCS = Request Communications Support

Figure 4.7-2. PRONG CSCI Context Diagram (cont.)

Table 4.7-3. PRONG CSCI Interface Events (1 of 3)

Event	Interface Event Description
Request Product	The PRONG CSCI sends requests, to the **SDSRV CSCI**, for particular data granules to be pushed, via the FTP service, onto the DPS science processor as input for data processing or for SSIT work.
Acquire PGE tar file	The PRONG CSCI acquires a tar file for any PGE not currently local to the science processor from the **SDSRV CSCI**. The executable is extracted from the tar file and used during PGE execution.
Insert Product History Tar Files	The PRONG CSCI sends a request to the **SDSRV CSCI** to insert the PGE Production History Tar File resulting outputs for permanent archive after the PGE has successfully completed executing.
Insert Failed PGE Tar File	After an unsuccessful execution of a PGE, the PRONG CSCI obtains the Tar file containing the PGE log files, core dump (if any), PCF and other files, and requests the files be inserted into the **SDSRV CSCI** for permanent archive.
Update Product metadata	The Operations Staff uses the QA Monitor GUI in the PRONG CSCI to send requests to update product metadata in the **SDSRV CSCI**.
Retrieve Product History	The Operations Staff uses the QA Monitor GUI to submit requests to the **SDSRV CSCI** to transfer the Production History tar file from the Science Data archives to the user's host machine.
Request Granule Deletion	The PRONG CSCI sends delete requests to the **SDSRV CSCI** for particular granules (interim data) in the archive and associated metadata to be deleted from the SDSRV inventory.
Request Data Insert	The PRONG CSCI sends requests to the **SDSRV CSCI** to insert a particular file or files into the archive, and catalog the associated metadata in the SDSRV inventory. These files can be processing output, static files received with PGEs, PGE tar files, APs, SSAPs or DAPs, failed PGE tar files, or production history files.
Request MCF	The PRONG CSCI requests a Metadata Configuration File (MCF) template for each output data type for specific PGEs from the **SDSRV CSCI** and the MCF template is populated with metadata from the output granule.
Query Data	The PRONG CSCI submits requests of this type to the **SDSRV CSCI**. It searches the archive for granules that match the user-supplied selection criteria: data type and begin/end date. Results are displayed to the user.
Retrieve Data	The PRONG CSCI sends retrieval requests, to the **SDSRV CSCI**, for a particular data granuleId. The product is transferred (pushed), via the File Transfer Protocol (FTP) service, onto the DPS science processor and used as input for Product Generation Executive (PGE) processing or for Science Software Integration and Test (SSIT) work.
Return Product	The data granules requested by the PRONG CSCI are sent from the **SDSRV CSCI**.
Return PGE Tar File	After an unsuccessful execution of a PGE, the PRONG CSCI obtains the Tar file containing the PGE log files, core dump (if any), PCF and other files, and requests the files be inserted into the **SDSRV CSCI** for permanent archive.

Table 4.7-3. PRONG CSCI Interface Events (2 of 3)

Event	Interface Event Description
Return MCF Template	The PRONG CSCI receive the MCF template to populate, from the **SDSRV CSCI**, as part of the GetMCF service call.
Return ESDT	The SDSRV CSCI returns the requested ESDT to the PRONG CSCI.
Create DPR	The PRONG CSCI uses the dprId from the **PLANG CSCI** to insert a job box for a DPR into AutoSys if all the input data required for the DPR is available. If the input data is not available, information used to construct the job in AutoSys is queued by the Job Management CSC until a release DPR is received.
Release DPR	The PRONG CSCI uses the dprId from the **PLANG CSCI** to release jobs currently waiting for external data in the Job Management queue into AutoSys.
Cancel DPR	The PRONG CSCI uses the dprId from the **PLANG CSCI** to delete jobs in AutoSys or from the queue.
Create Ground Event Job	The PRONG CSCI uses the ground event Id from the **PLANG CSCI** to create a ground event job in AutoSys to perform maintenance activities on data processing resources.
Cancel Ground Event Job	The PRONG CSCI uses the ground event Id from the **PLANG CSCI** to delete a ground event job in AutoSys.
DPR Control, Status, & Q/A	The **M&O staff** controls Data Processing Request (DPR) activity with the capability to cancel, suspend, resume, and modify a DPR. The M&O staff supports status collecting, PRONG hardware resource monitoring and Quality Assurance validating processes.
Delete Granules	The **M&O staff** sends requests to the PRONG CSCI to delete granules associated with cancelled DPRs.
Change DPR ID	The **M&O staff** can change the DPR ID of an existing DPR.
Change Max Concurrent Jobs for PGE Limits	The **M&O staff** can add to or update values in the DpPrPgeLimits database table.
Change Min/Max DPRs for Job Class	The **M&O staff** can add to or update values in the DpPrClassSchedulingLimits database table.
On-Demand Failure Message	The PRONG CSCI informs the **ECS user** (the initiator of the request) if an On-Demand Processing Request fails, if such failure is unrecoverable.
Request Management Services	The **MCI** provides a basic management library of services to the CSCIs, implemented as client or server applications, using the DCCI CSCI Process Framework. The basic management library of services includes: • **System startup and shutdown -** Please refer to the release-related, current version of the Mission Operations Procedures for the ECS Project document (611) and the current ECS Project Training Material document (625), identified in Section 2.2.1 of this document. • **Update Request Status** - The PRONG CSCI informs the **MCI** to update the status of an On-Demand Processing Request, when such request changes status (i.e., from Running to Completed or from Running to Failure).

Table 4.7-3. PRONG CSCI Interface Events (3 of 3)

Event	Interface Event Description
Request Communications Support	The **DCCI CSCI** provides a library of services available to each SDPS and CSMS CSCI. The CSCI services required to perform specific assignments are requested from the DCCI CSCI. These services include: • CCS Middleware Support • Database Connection Services • Network & Distributed File Services • Bulk Data Transfer Services • Name/Address Services • Password Services • Server Request Framework (SRF) • Universal Reference (UR) • Error/Event Logging • Fault Handling Services • Mode Information • Query Registry - Retrieving the requested configuration attribute-value pairs from the Configuration Registry
Import Location Information	The PRONG CSCI requests server location information from the **DCCI CSCI** (CCS Name Server).
Return Configuration Parameters	The **DCCI CSCI** returns the requested configuration parameters to the PRONG CSCI.
Export Location Information	The CCS Middleware CSC stores physical and logical location information received from PRONG CSCI in the **DCCI CSCI** (CCS Name Server).

4.7.1.3 Processing Architecture

Figure 4.7-3 is the PRONG CSCI architecture diagrams. The diagrams show the events sent to the PRONG CSCI processes and the events the PRONG CSCI processes send to other processes. The PRONG CSCI consists of COTS software and ECS developed processes.

The PRONG CSCI has interfaces with:

- Planning Subsystem – The PLS creates a production plan executed by the PRONG CSCI through the use of DPRs. Each DPR represents one processing job performed by a DPS computer resource. The PRONG CSCI provides DPR status information to the PLS to assist in production management activities.

- Data Server Subsystem – The PRONG CSCI supports SDPS data generation by requesting and receiving data (Data Staging) from a Data Server maintaining raw data and generated products. Also, the PRONG CSCI transfers data (Data de-staging) to a Data Server to archive generated data products.

- SDP Toolkit – The PRONG CSCI provides the location of input data and the location for the generated output data products. While a PGE is executing, the PRONG CSCI monitors the execution and provides current status to the M&O staff. Status includes current processing event history (e.g., data staging, and execution). Process monitoring includes checking resource usage by the PGE. At PGE execution completion, the PRONG CSCI initiates the transfer of the generated data product to the respective Data Server.

- System Management Subsystem – The PRONG CSCI relies on the MSS services for resource management and thus provides system management information including fault, accounting, configuration, security, performance, and accountability to the MSS. It also uses the Request Status Tracking capability to update the status of On-Demand Requests made by users.
- Operations – Supports PGE execution management and monitoring and the generation of SDPS Data Products via a Human Machine Interface (HMI). The HMI supports status information collection for a DPR, controlling DPR executions, and monitoring the status of the DPS hardware resources. The HMI also supports manual quality assurance activities performed at the DAAC.

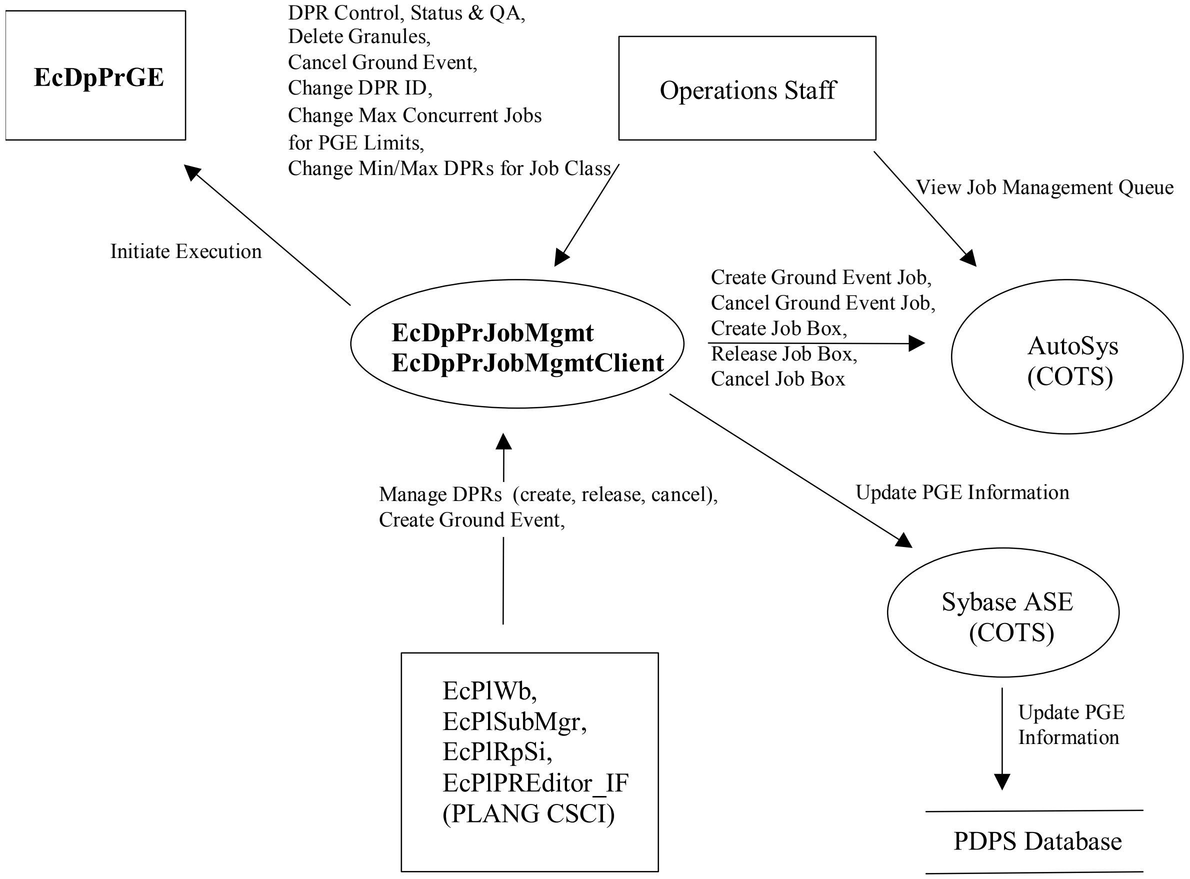


Figure 4.7-3. PRONG CSCI Architecture Diagram

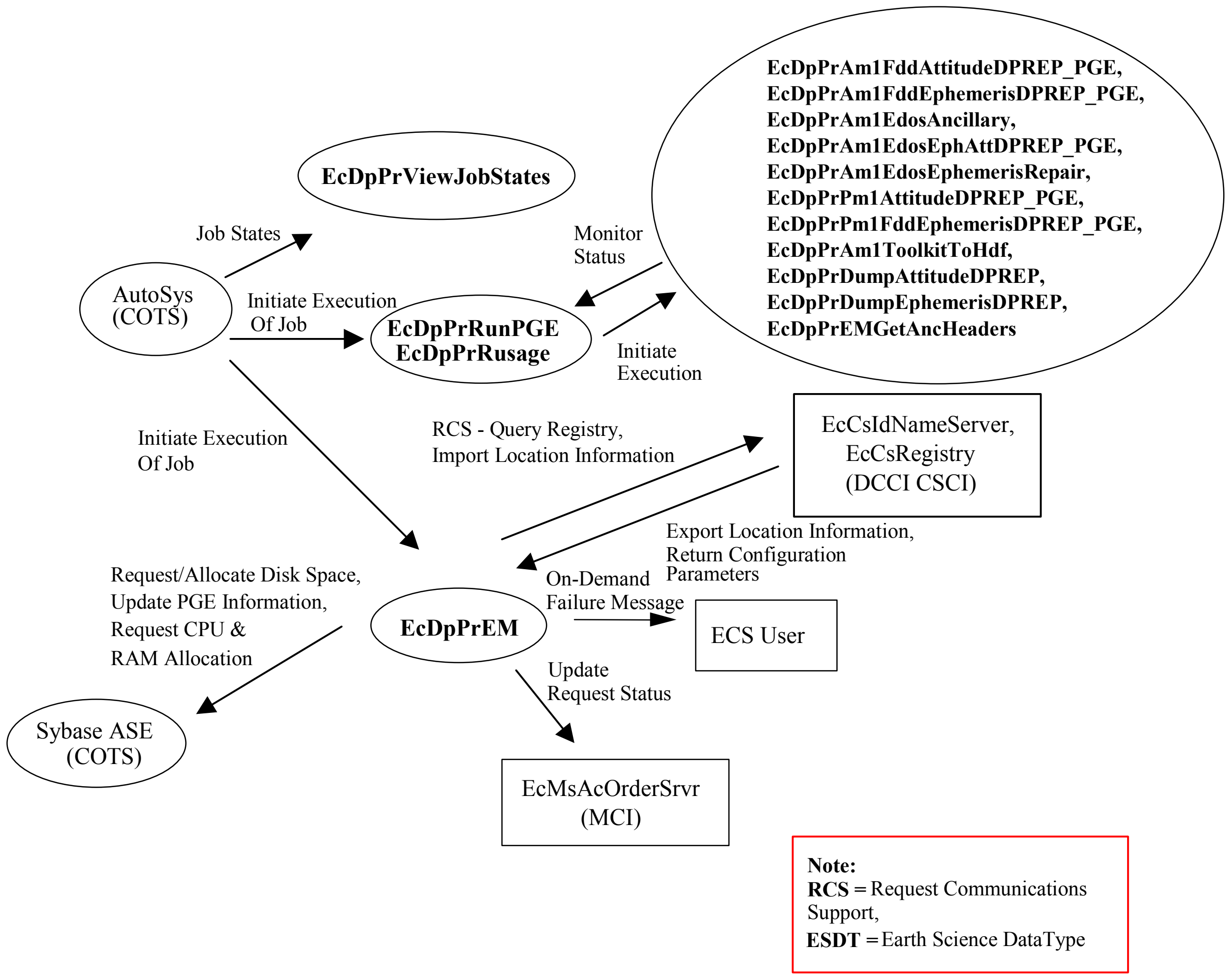

EcDpPrAm1FddAttitudeDPREP_PGE,
EcDpPrAm1FddEphemerisDPREP_PGE,
EcDpPrAm1EdosAncillary,
EcDpPrAm1EdosEphAttDPREP_PGE,
EcDpPrAm1EdosEphemerisRepair,
EcDpPrPm1AttitudeDPREP_PGE,
EcDpPrPm1FddEphemerisDPREP_PGE,
EcDpPrAm1ToolkitToHdf,
EcDpPrDumpAttitudeDPREP,
EcDpPrDumpEphemerisDPREP,
EcDpPrEMGetAncHeaders
EcDpPrViewJobStates
Job States
Monitor Status
AutoSys (COTS)
Initiate Execution Of Job
EcDpPrRunPGE EcDpPrRusage
Initiate Execution
Initiate Execution Of Job
RCS - Query Registry, Import Location Information
EcCsIdNameServer, EcCsRegistry (DCCI CSCI)
Export Location Information, Return Configuration Parameters
Request/Allocate Disk Space, Update PGE Information, Request CPU & RAM Allocation
EcDpPrEM
On-Demand Failure Message
ECS User
Update Request Status
Sybase ASE (COTS)
EcMsAcOrderSrvr (MCI)
Note:
RCS = Request Communications Support,
ESDT = Earth Science DataType

Figure 4.7-3. PRONG CSCI Architecture Diagram (cont.)

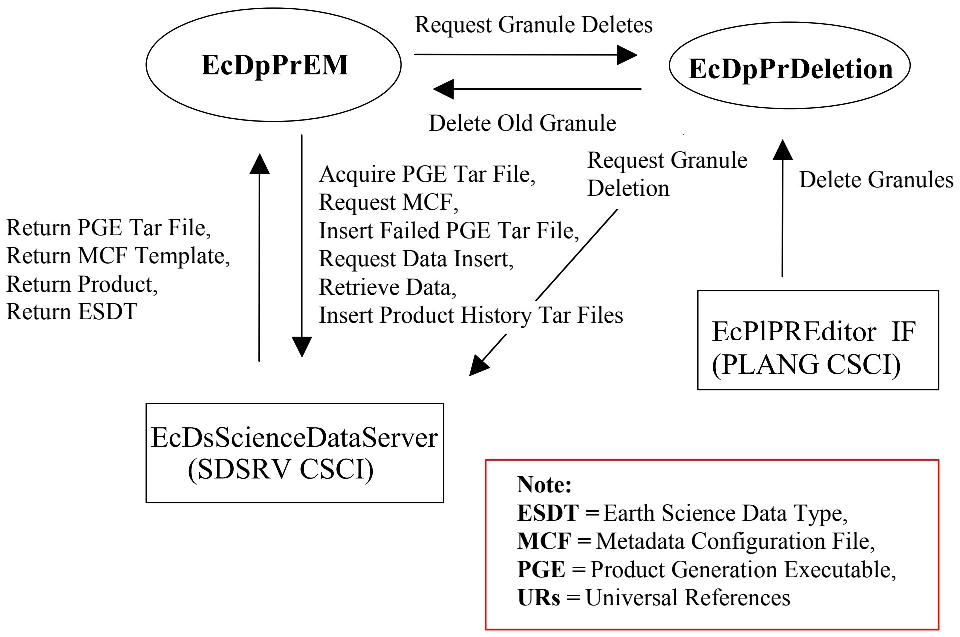

EcDpPrEM
Request Granule Deletes
Delete Old Granule
EcDpPrDeletion
Return PGE Tar File,
Return MCF Template,
Return Product,
Return ESDT
Acquire PGE Tar File,
Request MCF,
Insert Failed PGE Tar File,
Request Data Insert,
Retrieve Data,
Insert Product History Tar Files
Request Granule Deletion
Delete Granules
EcPlPREditor IF (PLANG CSCI)
EcDsScienceDataServer (SDSRV CSCI)
Note:
ESDT = Earth Science Data Type,
MCF = Metadata Configuration File,
PGE = Product Generation Executable,
URs = Universal References

Figure 4.7-3. PRONG CSCI Architecture Diagram (cont.)

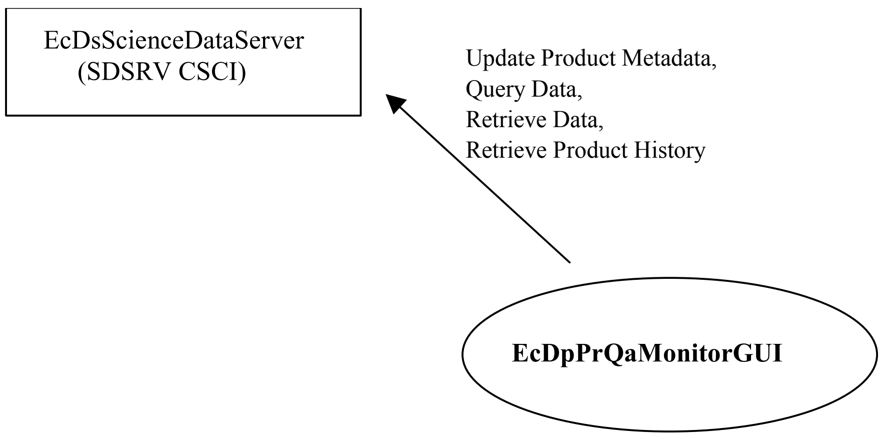


Figure 4.7-3. PRONG CSCI Architecture Diagram (cont.)

4.7.1.4 Processing Process Descriptions

Table 4.7-4 provides descriptions of the processes shown in the PRONG CSCI architecture diagrams. **Note:** Resource Management and the data manager are an integral part of PRONG. Unlike the Execution Manager, they are libraries of routines used by the Execution Manager instead of being executed like it. The Resource Manager checks the resource requirements (i.e., disk space, memory, and Central Processing Unit (CPU)) of a PGE during processing, determines the availability of those resources, and dedicates their usage until PGE execution finishes. The Data Management library (DpPrDM) manages the flow of science data to and from science processing resources. Data Management manages data retention on science processing resources to support PGE executions.

DPREP Background Information

A Flight Dynamics Division (FDD) data set consists of a couple of headers followed by a number of records. Each FDD record has fifty ephemeris points, spaced in time by a spacing, which is mentioned in the first FDD header. Each ephemeris point consists of time, three position coordinates, three velocity coordinates and a quality flag. The quality flag is set by DPREP. The rest comes in the FDD data set. Each FDD data set will have two hours of data for Terra (AM-1) and 24 hours worth of data for Aqua (PM-1) and Aura.

DPREP must read in all the FDD records and look for node crossings. A node crossing occurs when the satellite crosses the earth's equatorial plane. If the crossing is from south to north, it is called an ascending node crossing. If the crossing is from north to south it is called a descending node crossing.

The raw data comes in the form of position and velocity coordinates. The position transition from one point to another has to be checked for an equator crossing and the type and longitude of the crossing computed. Orbits are defined as being from one ascending node to another and are sequentially numbered from the first orbit. In addition, it is mandated for all orbits touched, orbit metadata (which is defined as the orbit number, the ascending and descending node crossing times and the descending node crossing longitude) has to be computed and appended at the end of the processed ephemeris data.

In addition, QA analysis must be performed. There are mainly two kinds of QA: data gaps and range violations. Data gaps correspond to missing data in the file or between data sets. A range violation is position or velocity magnitudes, which are less than a critical minimum or more than a critical maximum. This QA is written out in the header of the output HDF and native format data sets and also in the metadata files.

For a description of DPREP functionality, refer to the following documents:

Terra Spacecraft Ephemeris and Attitude Data Processing (Document Number 184-TP-001-002)

EOSDIS Spacecraft Ephemeris and Attitude Data Specification: Contents and Structure (available on the World Wide Web at http://newsroom.gsfc.nasa.gov/sdptoolkit/appendl.html)

Release 6A.07 SDP Toolkit User's Guide for the ECS Project, Appendix L (document number 333-CD-605)

The list of available documents will grow as more platforms are brought on-line for which DPREP must process ephemeris and attitude data.

Table 4.7-4. PRONG CSCI Processes (1 of 4)

Process	Type	COTS/ Developed	Functionality
EcDpPrEM	Other	Developed	The Execution Management process initiates the execution of PGEs (via the COTS product AutoSys). EcDpPrEM supports the preparation activities prior to the execution of PGEs and subsequent activities to the execution of PGEs. EcDpPrEM also provides status on On-Demand Processing Requests and send out e-mail to the originator in the event of a failure. The Data Management library portion of EcDpPrEM (DpPrDM) manages the flow of science data to and from science processing resources including communication mechanisms to interface with the EcDsScienceDataServer. Data Management manages data retention on science processing resources to support PGE executions.
EcDpPrRunPGE EcDpPrRusage	Other	Developed	The PGE Execution Manager process controls and monitors PGE executions including Process Control File creation and output product storage growth. EcDpPrRusage measures the actual resources used by the PGE and reports to AutoSys unexpected resource usage.
EcDpPrDeletion	Server	Developed	This CCS Middleware notifies the EcDsScienceDataServer to remove interim granules via the execution management process (EcDpPrEM) when they are no longer needed. The interim products are removed after the last PGE in the chain has used them or a pre set time has expired after the last use of the interim product. It also is used by the PLS to delete granules associated with a cancelled DPR.

Table 4.7-4. PRONG CSCI Processes (2 of 4)

Process	Type	COTS/ Developed	Functionality
AutoSys	GUI	COTS	AutoSys is a job scheduling software application used to automate operations in a distributed UNIX environment. AutoSys executes jobs to automate support for PGE execution. AutoSys creates job boxes consisting of a series of related jobs, and manages job dependencies. AutoSys provides graphical depictions of completed jobs and jobs being processed. It includes the Operator Console GUI to allow human intervention into monitoring and altering the AutoSys job stream. The daily job schedule is submitted to the Job Management server at the start of the processing day. Jobs, which have data available, are released into AutoSys. To support job executions, AutoSys requires additional help for: • Allocation of sufficient resources (e.g., disk space) to support executions. The EcDpPrEM provides the capabilities to manage disk space and monitor resources • Managing remote host data acquisition, data retention on the DPS processing host, and data distribution from the DPS processing host • Initialization and PGE executions.

Table 4.7-4. *PRONG CSCI Processes (3 of 4)*

Process	Type	COTS/ Developed	Functionality
EcDpPrJobMgmt EcDpPrJobMgmtClient EcDpPrViewJobStates	Server	Developed	The Job Management process uses the AutoSys COTS product to create and initiate execution of PRONG administrative jobs for managing SPRHW assets and for PGE execution. Three Unix processes bundled together into an AutoSys job box perform this work. Job Management is responsible for efficient AutoSys management so the maximum number of jobs possible can be continuously run using the product. This involves controlling the flow of jobs through AutoSys by only allowing jobs ready to run into the product and by removing jobs as they complete. Job Management also creates and starts execution of Ground Event jobs in AutoSys. The Job Management Client process is used by programs that need access to the Job Management Server services to modify jobs in AutoSys to change the priority of the jobs. The various events this process provides are: CreateDPR: A data processing request identified is then translated into seven standard process steps (one, the PGE execution, the remaining performing support activities). If the science data is available, the job box containing the seven individual jobs is released into AutoSys. ReleaseDPR: A previously created data processing request waiting for the availability of science data is released into AutoSys to begin execution. Change DPR Id: Rename a DPR Change Max Concurrent Jobs for PGE Limits Table: Modify DpPrPgeLimits database table. Change Min/Max DPRs for Job Class: Modify DpPrClassSchedulingLimits database table. CancelDPR: This provides the capability to cancel/terminate a data processing request. CreateGEvntJob: Create a Ground Event Job in AutoSys. CancelGEvntJob: Cancel a Ground Event Job. ViewJobManagement DPR Queue: The View Job States process allows the Operations Staff to view jobs in the queue to determine the completed jobs, the jobs executing, and the jobs awaiting execution.

Table 4.7-4. PRONG CSCI Processes (4 of 4)

Process	Type	COTS/ Developed	Functionality
EcDpPrQaMonitorGUI	GUI	Developed	This process provides the capability to transfer science data from the archives, browse data images, and examine and update science metadata. It is an automated tool for performing data analysis in support of DAAC Quality Assurance activities.
EcDpPrAm1EdosAncillary, EcDpPrAm1EdosEphAttDPREP_PGE, EcDpPrAm1EdosEphemerisRepair, EcDpPrAm1FddAttitudeDPREP_PGE, EcDpPrAm1FddEphemerisDPREP_PGE, EcDpPrAm1ToolkitToHdf, EcDpPrDumpAttitudeDPREP, EcDpPrDumpEphemerisDPREP, EcDpPrPm1AttitudeDPREP_PGE, EcDpPrPm1FddEphemerisDPREP_PGE, EcDpPrEMGetAncHeaders, EcDpPrAuraEphemerisDPREP_PGE, EcDpPrAuraAttitudeDPREP_PGE	Other	Developed	Data Preprocessing manages attitude and ephemeris data preprocessing for inputs to PGEs. Data preprocessing is the preliminary processing or application of an operation on a data set that does not alter or modify scientific content of the data set. Preprocessing includes data set format changes by reordering the lower level byte structure, data set reorganization (ordering data items within and between physical files), and preparing additional metadata based on lower level metadata. PM1 and Aura Orbit Processing include reformatting the FDD definitive ephemeris data sets into the toolkit native format and the HDF format, respectively. DPREP generates the orbit metadata records for the data sets. AM1 DPREP processes L0 ancillary data to produce ephemeris and attitude files formatted for use by the toolkit and in HDF. PM1 and Aura uses "carry-out" attitude information and FDS generated ephemeris to produce attitude files formatted for use by the Toolkit and in HDF format.
Sybase ASE	Server	COTS	The Sybase ASE acts as a SQL server for the PDPS database.
EcDpPrGE	Other	Developed	The EcDpPrJobMgmt server initiates the EcDpPrGE when the server gets a ground event request. The ground event process starts at a specified time and runs a specified duration. During the time the ground event process runs, it sets a computer resource (cpu, ram, etc.) off-line and the computer resource is not available for PGEs.

4.7.1.5 Processing Process Interface Descriptions

Table 4.7-5, provides descriptions of the interface events shown in the PRONG CSCI architecture diagrams.

Table 4.7-5. PRONG CSCI Process Interface Events (1 of 13)

Event	Event Frequency	Interface	Initiated By	Event Description
Request Management Services	At system startup or shutdown and for restarts	*Processes:* EcDpPrJobMgmt, EcDpPrJobMgmtClient	DAAC unique startup scripts	**System startup and shutdown -** Please refer to the release-related, current version of the Mission Operations Procedures for the ECS Project document (611) and the current ECS Project Training Material document (625), identified in Section 2.2.1 of this document.
DPR Control, Status & QA	Per data processing request	*Process:* EcDpPrJobMgmt *Library:* DpPrJM *Class:* DpPrScheduler	Operations Staff/ Operations Staff terminal	The **Operations staff** controls Data Processing Request (DPR) activity with the capability to cancel, suspend, resume, and modify a DPR. The M&O staff supports status collecting, PRONG hardware resource monitoring and Quality Assurance validating processes.
Delete Granules	Per operations request	*Processes:* EcDpPrJobMgmt EcDpPrJobMgmtClient	M&O staff	The **M&O staff** sends requests to the EcDpPrJobMgmt/EcDpPrJobMgmtClient to delete granules associated with cancelled DPRs.
Cancel Ground Event	Per operations request	*Processes:* EcDpPrJobMgmt EcDpPrJobMgmtClient	M&O staff	The **M&O staff** sends a request to the EcDpPrJobMgmt process to cancel ground events in AutoSys.
Change DPR ID	One per defined DPR	*Processes:* EcDpPrJobMgmt EcDpPrJobMgmtClient	M&O staff	The **M&O staff** can change the DPR ID of an existing DPR.
Change Max Concurrent Jobs for PGE Limits Table	Per Operations Staff Request	*Process:* EcDpPrJobMgmt EcDpPrJobMgmtClient	M&O staff	The **M&O staff** can add to or update values in the DpPrPgeLimits database table.
Change Min/Max DPRs for Job Class	Per operations staff request	*Process:* EcDpPrJobMgmt EcDpPrJobMgmtClient	M&O staff	The **M&O staff** can add to or update values in the DpPrClassSchedulingLimits database table.

Table 4.7-5. PRONG CSCI Process Interface Events (2 of 13)

Event	Event Frequency	Interface	Initiated By	Event Description
View Job Management DPR Queue	Per Operations Staff request.	*Process:* EcDpPrViewJobStates *Class:* DpPrListJobs	Operations Staff/ Operations Staff terminal	The **Operations staff** can view the job state via the EcDpPrViewJobStates process, as an aid in scheduling jobs.
Create Ground Event Job	One per defined ground event	*Process:* AutoSys (COTS)	*Process:* EcDpPrJobMgmt *Library:* DpPrJM *Class:* DpPrScheduler	The EcDpPrJobMgmt process sends a request to **AutoSys** to create a ground event job to perform maintenance activities on data processing resources.
Cancel Ground Event Job	One per defined ground event	*Process:* AutoSys (COTS)	*Process:* EcDpPrJobMgmt *Library:* DpPrJM *Class:* DpPrScheduler	The EcDpPrJobMgmt process sends a request to **AutoSys** to cancel a ground event job for the deletion of a ground event.
Create Job Box	One per DPR	*Process:* AutoSys (COTS)	*Process:* EcDpPrJobMgmt *Library:* DpPrJM *Class:* DpPrScheduler	The EcDpPrJobMgmt sends a request to **AutoSys** to create a job box when all DPR input data is available using Job Interface Language (JIL). If all the input data is not available, the EcDpPrJobMgmt stores the DPR in a priority-based queue.
Release Job Box	One per execution of a DPR in the job mgmt queue	*Process:* AutoSys (COTS)	*Process:* EcDpPrJobMgmt *Library:* DpPrJM *Class:* DpPrScheduler	The EcDpPrJobMgmt sends a request to release (or delete) jobs box in the **AutoSys** queue for a DPR awaiting execution.
Cancel Job Box	One per job box deletion	*Process:* AutoSys (COTS)	*Process:* EcDpPrJobMgmt *Library:* DpPrJM *Class:* DpPrScheduler	The EcDpPrJobMgmt deletes a job box in **AutoSys** and performs any cleanup when an operator requests a DPR to be canceled.

Table 4.7-5. *PRONG CSCI Process Interface Events (3 of 13)*

Event	Event Frequency	Interface	Initiated By	Event Description
Update PGE Information	One per update/retrieve data request	Sybase ASE Database (COTS)	*Processes:* EcDpPrJobMgmt, EcDpPrJobMgmtClient *Library:* DpPrRM	The EcDpPrJobMgmt and EcDpPrJobMgmtClient processes, using the DpPrRM library, request update and retrieval of data in the **Sybase ASE** database that defines a PGE and allows the PGE to be scheduled and executed by AutoSys. These processes request updates for granule information (location, size, etc.), processing status, and check-pointing information stored in the database.
Manage DPRs (create, release, cancel)	One per control request	*Process:* EcDpPrJobMgmt *Library:* DpPrJM *Class:* DpPrScheduler	*Processes:* EcPlWb, EcPlSubMgr, EcPlRpSi *Library:* PlCore1 *Classes:* PlWbScheduler, PlDpr	The **EcPlWb** process sends requests to the EcDpPrJobMgmt to create and cancel DPR jobs in AutoSys. The EcPlPREditor_IF process sends requests to delete DPRs in AutoSys to the EcDpPrJobMgmt process. The EcPlSubMgr process sends requests to the EcDpPrJobMgmt process to release DPR jobs in AutoSys.
Create Ground Event	One per ground resource	*Process:* EcDpPrJobMgmt *Library:* DpPrJM *Class:* DpPrScheduler	*Process:* EcPlWb *Class:* PlWbScheduler	The **EcPlWb** sends requests to the EcDpPrJobMgmt process to create ground events for maintenance activities on data processing resources.
Cancel Ground Event	One per ground resource	*Process:* EcDpPrJobMgmt *Library:* DpPrJM *Class:* DpPrScheduler	*Process:* EcPlRpSi	The **EcPlRpSi** process sends a request to the EcDpPrJobMgmt process to cancel ground events in AutoSys.
Initiate execution (control)	One per PGE job execution	*Process:* EcDpPrGE	*Process:* EcDpPrJobMgmt *Library:* DpPrJM *Class:* DpPrScheduler	The EcDpPrJobMgmt process initiates the **EcDpPrGE** process when ground events occur.

Table 4.7-5. PRONG CSCI Process Interface Events (4 of 13)

Event	Event Frequency	Interface	Initiated By	Event Description
Request Management Services	At system startup or shutdown and for restarts	*Process:* EcDpPrDeletion	DAAC unique startup scripts	**System startup and shutdown -** Please refer to the release-related, current version of the Mission Operations Procedures for the ECS Project document (611) and the current ECS Project Training Material document (625), identified in Section 2.2.1 of this document.
Job States	Per AutoSys Status update.	*Process:* EcDpPrViewJobStates *Class:* DpPrListJobs	*Process:* AutoSys (COTS)	The **AutoSys** provides the job state (completed, executing, or in a queue to be executed) to the EcDpPrViewJobStates process.
Initiate execution of job (control)	One per PGE job execution	***Process:*** EcDpPrEM *Library:* DpPrEM *Class:* DpPrExecutionManager ***Process:*** EcDpPrRunPGE	Process: AutoSys (COTS)	**AutoSys** initiates the execution of the EcDpPrEM and the EcDpPrRunPGE processes to control the preparation (data staging), execution, and archiving of higher level products, which are produced, and cleanup of each PGE run.

Table 4.7-5. PRONG CSCI Process Interface Events (5 of 13)

Event	Event Frequency	Interface	Initiated By	Event Description
Monitor status (status)	One per PGE job execution	*Processes:* EcDpPrAm1EdosAncillary, EcDpPrAm1EdosEphAttDPREP_PGE, EcDpPrAm1EdosEphemerisRepair, EcDpPrAm1FddAttitudeDPREP_PGE, EcDpPrAm1FddEphemerisDPREP_PGE, EcDpPrAm1ToolkitToHdf, EcDpPrDumpAttitudeDPREP, EcDpPrDumpEphemerisDPREP, EcDpPrPm1FddEphemerisDPREP_PGE, EcDpPrEMGetAncHeaders, EcDpPrPm1AttitudeDPREP_PGE, EcDpPrAuraEphemerisDPREP_PGE, EcDpPrAuraAttitudeDPREP_PGE, Other PGE executables created by science teams	*Process:* EcDpPrRunPGE	The EcDpPrRunPGE process, apart from initiating the PGE process, also monitors the PGE's computer resources. If the PGE's computer resources exceed its expected usage an alarm is sent to the AutoSys. The **EcDpPrAm1EdosAncillary**, **EcDpPrAm1EdosEphAttDPREP_PGE**, **EcDpPrAm1EdosEphemerisRepair**, **EcDpPrAm1FddAttitudeDPREP_PGE**, **EcDpPrAm1FddEphemerisDPREP_PGE**, **EcDpPrAm1ToolkitToHdf**, **EcDpPrDumpAttitudeDPREP**, **EcDpPrDumpEphemerisDPREP**, **EcDpPrPm1FddEphemerisDPREP_PGE**, **EcDpPrEMGetAncHeaders** and **EcDpPrPm1AttitudeDPREP_PGE**, **EcDpPrAuraEphemerisDPREP_PGE**, **EcDpPrAuraAttitudeDPREP_PGE** and other PGE processes send status to the EcDpPrRunPGE process.

Table 4.7-5. PRONG CSCI Process Interface Events (6 of 13)

Event	Event Frequency	Interface	Initiated By	Event Description
Initiate execution (control)	One per PGE job execution	*Processes:* EcDpPrAm1EdosAncillary, EcDpPrAm1EdosEphAttDPREP_PGE, EcDpPrAm1EdosEphemerisRepair, EcDpPrAm1FddAttitudeDPREP_PGE, EcDpPrAm1FddEphemerisDPREP_PGE, EcDpPrAm1ToolkitToHdf, EcDpPrDumpAttitudeDPREP, EcDpPrDumpEphemerisDPREP, EcDpPrPm1FddEphemerisDPREP_PGE, EcDpPrEMGetAncHeaders, EcDpPrPm1AttitudeDPREP_PGE, EcDpPrAuraEphemerisDPREP_PGE, EcDpPrAuraAttitudeDPREP_PGE, Other PGE executables created by science teams	*Process:* EcDpPrRunPGE	The EcDpPrRunPGE provides a buffer between AutoSys and the PGE. This serves as a wrapper to the PGE process, initiates the PGE execution and captures the PGE's exit status. The EcDpPrRunPGE initiates the **EcDpPrAm1EdosAncillary**, **EcDpPrAm1EdosEphAttDPREP_PGE**, **EcDpPrAm1EdosEphemerisRepair**, **EcDpPrAm1FddAttitudeDPREP_PGE**, **EcDpPrAm1FddEphemerisDPREP_PGE**, **EcDpPrAm1ToolkitToHdf**, **EcDpPrDumpAttitudeDPREP**, **EcDpPrDumpEphemerisDPREP**, **EcDpPrPm1FddEphemerisDPREP_PGE**, **EcDpPrEMGetAncHeaders**, **EcDpPrPm1AttitudeDPREP_PGE**, **EcDpPrAuraEphemerisDPREP_PGE,** and **EcDpPrAuraAttitude** processes.

Table 4.7-5. PRONG CSCI Process Interface Events (7 of 13)

Event	Event Frequency	Interface	Initiated By	Event Description
Request Communications Support	One service per request.	***Process:*** EcCsIdNameServer ***Libraries:*** EcPf, Middleware, FoNs, FoIp, oodce *Classes:* EcPfManagedServer, EcPfClient, CCSMdwNameServer, FoNsNameServerProxy, CCSMdwRwNetProxy ***Library (Common):*** EcUr *Class:* EcUrServerUR ***Library:*** event *Class:* EcLgErrorMsg ***Process:*** EcCsRegistry *Library:* EcCsRegistry *Class:* EcRgRegistryServer_C	*Process:* EcDpPrEM *Library:* DpPrDM *Class:* DpPrDataManager	The **DCCI CSCI** provides a library of services available to each SDPS and CSMS CSCI. The CSCI services required to perform specific assignments are requested from the DCCI CSCI. These services include: • CCS Middleware Support • Database Connection Services • Network & Distributed File Services • Bulk Data Transfer Services • Name/Address Services • Password Services • Server Request Framework (SRF) • Universal Reference (UR) • Error/Event Logging • Fault Handling Services • Mode Information • Query Registry - Retrieving the requested configuration attribute-value pairs from the Configuration Registry

Table 4.7-5. PRONG CSCI Process Interface Events (8 of 13)

Event	Event Frequency	Interface	Initiated By	Event Description
Import Location Information	As required for processing	*Process:* EcDpPrEM *Library:* DpPrDM *Class:* DpPrDataManager	*Process:* EcCsIdNameServer *Library:* EcPf, Middleware, FoNs, FoIp, oodce *Classes:* EcPfManagedServer, EcPfClient, CCSMdwNameServer, FoNsNameServerProxy, CCSMdwRwNetProxy	The EcDpPrEM requests server location information from the **EcCsIdNameServer**.
Export Location Information	Once at system start up and after each failure recovery	*Process:* EcCsIdNameServer *Library:* EcPf, Middleware, FoNs, FoIp, oodce *Classes:* EcPfManagedServer, EcPfClient, CCSMdwNameServer, FoNsNameServerProxy, CCSMdwRwNetProxy	*Process:* EcDpPrEM *Library:* DpPrDM *Class:* DpPrDataManager	The EcDpPrEM places physical and logical location information in the **EcCsIdNameServer**.
Return Configuration Parameters	One per configuration registry query	*Process:* EcDpPrEM *Library:* DpPrDM *Class:* DpPrDataManager	*Process:* EcCsRegistry *Library:* EcCsRegistry *Class:* EcRgRegistryServer_C	The **EcCsRegistry** returns the requested configuration parameters to the EcDpPrEM.

Table 4.7-5. PRONG CSCI Process Interface Events (9 of 13)

Event	Event Frequency	Interface	Initiated By	Event Description
On-Demand Failure Message	Only in the event of On-Demand Processing Request failure	User *Library:* CsEmMailRelA *Class:* CsEmMailRelA	*Process:* EcDpPrEM *Library:* DpPrEM *Class:* DpPrExecutionManager	The EcDpPrEM process sends an e-mail message to the **originator** (**a user**) of an On-Demand Processing Request if any DPR generated by that request fails.
Update Request Status	One for each PGE run to satisfy the requested On-Demand Product	*Process:* EcMsAcOrderSrvr *Libraries:* MsAcSrv, MsAcComm *Class:* EcAcOrderCMgr	*Process:* EcDpPrEM *Library:* DpPrEM *Class:* DpPrExecutionManager	The EcDpPrEM process updates the status of an On-Demand Processing Request, via the **EcMsAcOrderSrvr**, when a DPR for the order completes or fails processing.
Request/Allocate Disk Space	One per disk request	Sybase ASE (COTS)	*Process:* EcDpPrEM *Libraries:* DpPrRM, DpPrDM *Class:* DpPrResourceManager	The EcDpPrEM requests disk space via the **Sybase ASE** by using the DpPrRM and DpPrDM library software for each input granule that needs to be staged to the local processing disk and output granule needed by a PGE.
Update PGE Information	One per update/retrieve data request	Sybase Database (COTS)	*Process:* EcDpPrEM *Libraries:* DpPrRM, DpPrDM	The EcDpPrEM process, using the DpPrRM and DpPrDM libraries, requests update and retrieval of data in the **Sybase ASE** database that defines a PGE and allows the PGE to be scheduled and executed by AutoSys. These processes request updates for granule information (location, size, etc.), processing status, and check-pointing information stored in the database.
Request CPU and RAM Allocation	One per PGE job execution	Sybase ASE (COTS)	*Process:* EcDpPrEM *Libraries:* DpPrRM, DpPrDM *Class:* DpPrResourceManager	The EcDpPrEM process requests CPU and RAM allocations via the **Sybase ASE** by using the DpPrRM library software for each PGE based on values entered at SSIT.

Table 4.7-5. PRONG CSCI Process Interface Events (10 of 13)

Event	Event Frequency	Interface	Initiated By	Event Description
Request Granule Deletes	Per granule delete request	*Process:* EcDpPrDeletion *Class:* DpDeletion	*Process:* EcDpPrEM *Libraries:* DpPrDssIF, DpPrDM *Class:* DpPrDSSInterface	The EcDpPrEM sends requests to the EcDpPrDeletion process to delete interim granules a PGE had used in processing or after a defined storage period has elapsed. The EcDpPrDeletion process submits the request to the **EcDsScienceDataServer**.
Delete Old Granule	One per granule	*Process:* EcDpPrEM *Library:* DpPrDM *Class:* DpPrGranuleLocator	*Process:* EcDpPrDeletion *Class:* DpDeletion	The **EcDpPrDeletion** process gets a request from the EcPlWb, via the EcDpPrEM, to delete a granule from one of the local science processing disks. The EcDpPrDeletion process uses the DpPrDM library software to delete the files from the disk and update the PDPS database.
Delete Granules	Per granule delete request	*Process:* EcDpPrDeletion *Class:* DpDeletion	*Process:* EcPlPREditor_IF *Library:* PlCore1 *Classes:* PlWbScheduler, PlDpr	The **EcPlPREditor_IF** sends requests to the EcDpPrDeletion process to delete granules associated with a DPR, which has been cancelled.
Request Granule Deletion	Per granule delete request	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClRequest, DsClCommand, DsClESDTReferenceCollector	*Process:* EcDpPrDeletion *Class:* DpDeletion	The EcDpPrDeletion process submits the request to delete interim granules a PGE had used in processing or after a defined storage period has elapsed to the **EcDsScienceDataServer**.
Acquire PGE Tar file	One per tar file acquire	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClRequest, DsClCommand	*Process:* EcDpPrEM *Library:* DpPrDssIF *Class:* DpPrDSSInterface	The EcDpPrEM acquires a tar file for any PGE not currently local to the science processor from the **EcDsScienceDataServer**. The tar file is removed from the tape archive and used during PGE execution.

Table 4.7-5. PRONG CSCI Process Interface Events (11 of 13)

Event	Event Frequency	Interface	Initiated By	Event Description
Request MCF	One per MCF request	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClDescriptor	*Process:* EcDpPrEM *Library:* DpPrDssIF *Class:* DpPrDSSInterface	The EcDpPrEM requests a MCF template for each output data type for specific PGEs from the **EcDsScienceDataServer** and the MCF template is populated with metadata from the output granule.
Insert Failed PGE Tar file	One per unsuccessful PGE execution	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClRequest, DsClCommand	*Process:* EcDpPrEM *Library:* DpPrDssIF *Class:* DpPrDSSInterface	After an unsuccessful execution of a PGE, the EcDpPrEM obtains the Tar file containing the PGE log files, core dump (if any), PCF and other files, and requests the files be inserted into the **EcDsScienceDataServer** for permanent archive.
Request Data Insert	One granule per request	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClRequest, DsClCommand, DsClESDTReferenceCollector	*Process:* EcDpPrEM *Libraries:* DpPrDssIF, DpPrDM *Class:* DpPrDSSInterface	After the PGE has successfully completed executing, the EcDpPrEM sends insert requests for the **EcDsScienceDataServer** to store the output granules into the SDSRV inventory/archives.
Retrieve Data	One granule per request	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClRequest, DsClCommand, DsClESDTReferenceCollector	*Process:* EcDpPrEM *Libraries:* DpPrDssIF, DpPrDM *Class:* DpPrDSSInterface	The EcDpPrEM acquires the input granules needed by a PGE not currently local on a science processor from the **EcDsScienceDataServer** by sending a UR for each granule.

Table 4.7-5. PRONG CSCI Process Interface Events (12 of 13)

Event	Event Frequency	Interface	Initiated By	Event Description
Insert Product History Tar Files	One per successful PGE execution	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClRequest, DsClCommand	*Process:* EcDpPrEM *Libraries:* DpPrDssIF, DpPrDM *Class:* DpPrDSSInterface	After the PGE has successfully completed executing and archiving the resulting outputs, the EcDpPrEM requests the PGE Production History Tar file be inserted into the **EcDsScienceDataServer** for permanent archive.
Return PGE Tar File	One per request	*Process:* EcDpPrEM *Library:* DpPrDssIF *Class:* DpPrDSSInterface	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClRequest, DsClCommand	After an unsuccessful execution of a PGE, the EcDpPrEM obtains the Tar file containing the PGE log files, core dump (if any), Process Control File (PCF) and other files, and requests the files be inserted into the **EcDsScienceDataServer** for permanent archive.
Return MCF Template	One per set of external data received by ECS	*Process:* EcDpPrEM *Library:* DpPrDssIF *Class:* DpPrDSSInterface	*Process:* EcDsScienceDataServer *Library:* DsCl *Class:* DsClDescriptor	The **EcDsScienceDataServer** provides the MCF template as part of the GetMCF service call to the EcDpPrEM process.
Return Product	One product per request	*Process:* EcDpPrEM *Library:* DpPrDssIF *Class:* DpPrDSSInterface	*Process:* EcDsScienceDataServer *Library:* DsCl *Class:* DsClDescriptor	The data granules requested by the EcDpPrEM are sent from the **EcDsScienceDataServer**.
Update Product Metadata	One per metadata product update	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClCommand, DsClRequest, DsClESDTReferenceCollector	*Process:* EcDpPrQaMonitorGUI *Library:* DpPrQaMonitor *Class:* DpPrQAGranuleQaFlags	The EcDpPrQaMonitorGUI provides the operator with capabilities to update product metadata in the **EcDsScienceDataServer**.

Table 4.7-5. PRONG CSCI Process Interface Events (13 of 13)

Event	Event Frequency	Interface	Initiated By	Event Description
Query Data	One per query	*Process:* EcDsScienceDataServer *Library:* DsCl *Class:* DsClESDTReferenceCollector	*Process:* EcDpPrQaMonitorGUI *Library:* DpPrQaMonitor *Class:* DpPrQaDataGranule	The EcDpPrQaMonitorGUI submits requests of this type to the **EcDsScienceDataServer**. It searches the archive for granules that match the user-supplied selection criteria: data type and begin/end date. Results are displayed to the user.
Retrieve Data	One per request	*Process:* EcDsScienceDataServer *Library:* DsCl *Class:* DsClAcquireCommand	*Process:* EcDpPrQaMonitorGUI *Library:* DpPrQaMonitor *Class:* DpPrQaMonitor	The EcDpPrQaMonitorGUI submits requests of this type to the **EcDsScienceDataServer**. It transfers a granule from the Science Data archive to the user's host machine.
Retrieve Product History	One per request	*Process:* EcDsScienceDataServer *Library:* DsCl *Class:* DsClAcquireCommand	*Process:* EcDpPrQaMonitorGUI *Library:* DpPrQaMonitor *Class:* DpPrQaMonitor	The EcDpPrQaMonitorGUI submits requests of this type to the **EcDsScienceDataServer**. It transfers the Production History tar file from the Science Data archive to the user's host machine.

4.7.1.6 Processing Data Stores

Table 4.7-6 provides descriptions of the data stores shown in the PRONG CSCI architecture diagram.

Table 4.7-6. PRONG CSCI Data Stores

Data Store	Type	Functionality
PDPS database	Database	The PDPS database is replicated within the same site and holds the persistent data for PDPS. The persistent data includes, but is not limited to, resource information entered with the Resource Planning utilities, PGE and data type information entered at SSIT, Production Requests, Data Processing Requests and Data Granule information entered using the Production Request Editor and plan information entered using the Production Planning Workbench.

4.7.2 Algorithm Integration and Test Tools Software Description

4.7.2.1 Functional Overview

The DAAC Integration and Test (I&T) team to use the Algorithm Integration and Test Tools (AITTL):

- Retrieve science software and submit it for configuration control
- Compile and link the delivered source files
- Execute test cases
- Provide error diagnosis using interactive debuggers, and data viewers
- Collect resource metrics of CPU time, memory, and disk space to build the PGE Profile and thus enable the PLANG and PRONG CSCIs to execute the science software
- Update the system databases after the science software completes acceptance testing

The AITTL tools are in the following categories:

- Compilers, linkers, debuggers, and other development and operating system tools
- Tools for viewing science software documentation
- Tools for checking compliance of science software to Earth Science Data and Information System (ESDIS)-specified coding standards
- Code analysis tools (e.g., Forte Developer, ProDev Workshop)
- Data viewing tools (e.g., EOSView)
- Tools for comparing HDF files
- Tools for comparing Binary files
- Tools for providing executable profiles (to get a PGE performance profile)
- Tools to register the science software with the Planning and Data Processing Subsystems
- Tools to add and update Science Software Archive Packages (SSAPs) in the Data Server
- Tools for writing reports and maintaining the I&T logs
- Tools for checking Process Control Files and for prohibited functions
- Tools to display product metadata

For information on science software integration and test procedures, see Science User's Guide and Operations Procedures Handbook for the ECS Project (205-CD-002-001) Part 4, and Science Software Integration and Test (JU9403V1). For information on the ESDIS science software coding standards and guidelines, see Data Production Software and Science Computing Facility (SCF) Standards and Guidelines (423-16-01).

Note: The directory structure for the AITTL software has the name SSIT and not AITTL. The use of the SSIT directory structure name is to denote the main purpose of the

Algorithm Test Tools as tools to support the Science Software Integration and Test activities as part of the SDPS data processing.

4.7.2.2 Algorithm Integration and Test Tools Context

Figure 4.7-4 is the Algorithm Integration and Test Tool (AITTL) CSCI context diagram. The diagram shows the events sent to the AITTL CSCI and the events the AITTL CSCI sends to other ECS subsystems. **Note: System startup and shutdown -** Please refer to the release-related, current version of the Mission Operations Procedures for the ECS Project document (611) and the current ECS Project Training Material document (625), identified in Section 2.2.1 of this document.

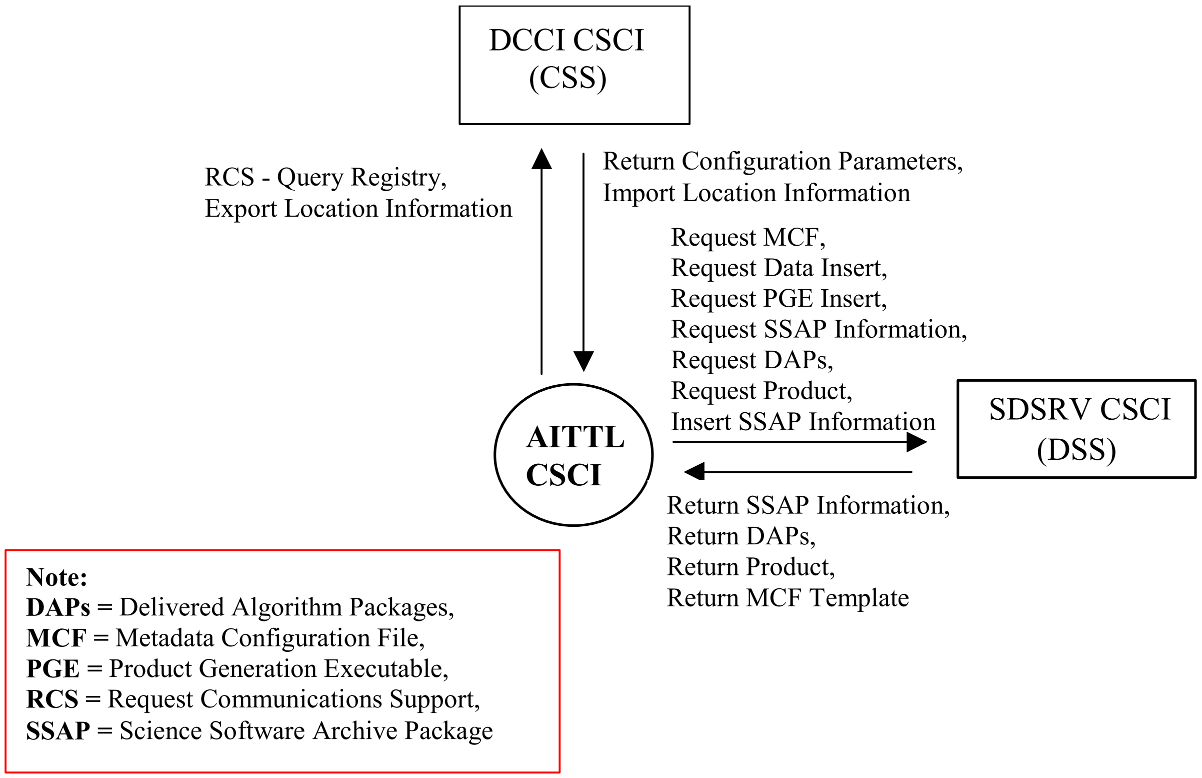

DCCI CSCI (CSS)
RCS - Query Registry, Export Location Information
Return Configuration Parameters, Import Location Information
Request MCF, Request Data Insert, Request PGE Insert, Request SSAP Information, Request DAPs, Request Product, Insert SSAP Information
SDSRV CSCI (DSS)
AITTL CSCI
Return SSAP Information, Return DAPs, Return Product, Return MCF Template
Note:
DAPs = Delivered Algorithm Packages,
MCF = Metadata Configuration File,
PGE = Product Generation Executable,
RCS = Request Communications Support,
SSAP = Science Software Archive Package

Figure 4.7-4. AITTL Context Diagram

Table 4.7-7 provides descriptions of the interface events shown in the AITTL Context Diagram.

Table 4.7-7. AITTL Interface Events (1 of 2)

Event	Interface Event Description
Return Configuration Parameters	The **DCCI CSCI** returns the requested configuration parameters to the AITTL CSCI.
Import Location Information	The AITTL CSCI requests server location information from the **DCCI CSCI** (CCS Name Server).
Request MCF	The AITTL CSCI sends requests to the **SDSRV CSCI** for the MCF template for use during SSIT. The PRONG CSCI also requests the MCF template from the SDSRV CSCI prior to a data insert request.
Request Data Insert	The AITTL CSCI puts various types of data into the SDSRV inventory (**SDSRV CSCI**), from SSAP information to Static files and PGE executables. In response, the AITTL CSCI gets the results of the insert and the UR to perform an acquire request.
Request PGE Insert	The AITTL CSCI sends requests to the **SDSRV CSCI** to insert data that defines a PGE and allows it to be scheduled and executed.
Request SSAP Information	The AITTL CSCI sends requests to the **SDSRV CSCI** for SSAP information, including names of existing SSAPs and the information associated with a specific SSAP. In response, the SDSRV CSCI sends lists of SSAPs and related information.
Request DAPs	The AITTL CSCI requests DAPs based on URs from the **SDSRV CSCI**. The requested DAPs are placed on a local AITTL disk.
Request Product	The AITTL CSCI sends requests, to the **SDSRV CSCI**, for particular data granules to be pushed, via the FTP service, onto the DPS science processor as input for data processing or for SSIT work.
Insert SSAP Information	The M&O Staff sends requests to the **SDSRV CSCI** to insert SSAP information, via the SSAP GUI, including SSAP name, SSAP version number, PGE name, PGE version number, and SSAP Acceptance Date.
Return SSAP Information	The **SDSRV CSCI** sends lists of SSAPs and related information to the AITTL CSCI.
Return DAPs	The **SDSRV CSCI** places the DAPs on a local AITTL CSCI disk.
Return Product	The data granules requested by the AITTL CSCI are sent from the **SDSRV CSCI**.
Return MCF Template	The AITTL CSCI receives the MCF template to populate, from the **SDSRV CSCI**, as part of the GetMCF service call.

Table 4.7-7. AITTL Interface Events (2 of 2)

Event	Interface Event Description
Request Communications Support	The **DCCI CSCI** provides a library of services available to each SDPS and CSMS CSCI. The CSCI services required to perform specific assignments are requested from the DCCI CSCI. These services include: • CCS Middleware Support • Database Connection Services • Network & Distributed File Services • Bulk Data Transfer Services • Name/Address Services • Password Services • Server Request Framework (SRF) • Universal Reference (UR) • Error/Event Logging • Fault Handling Services • Mode Information • Query Registry - Retrieving the requested configuration attribute-value pairs from the Configuration Registry
Export Location Information	The CCS Middleware CSC stores physical and logical location information received from the AITTL CSCI in the **DCCI CSCI** (CCS Name Server).

4.7.2.3 Algorithm Integration and Test Tools Architecture

Figure 4.7-5 is the AITTL CSCI architecture diagrams. The diagrams show the events that launch the AITTL CSCI processes and the events the AITTL CSCI processes send to processes in other CSCIs.

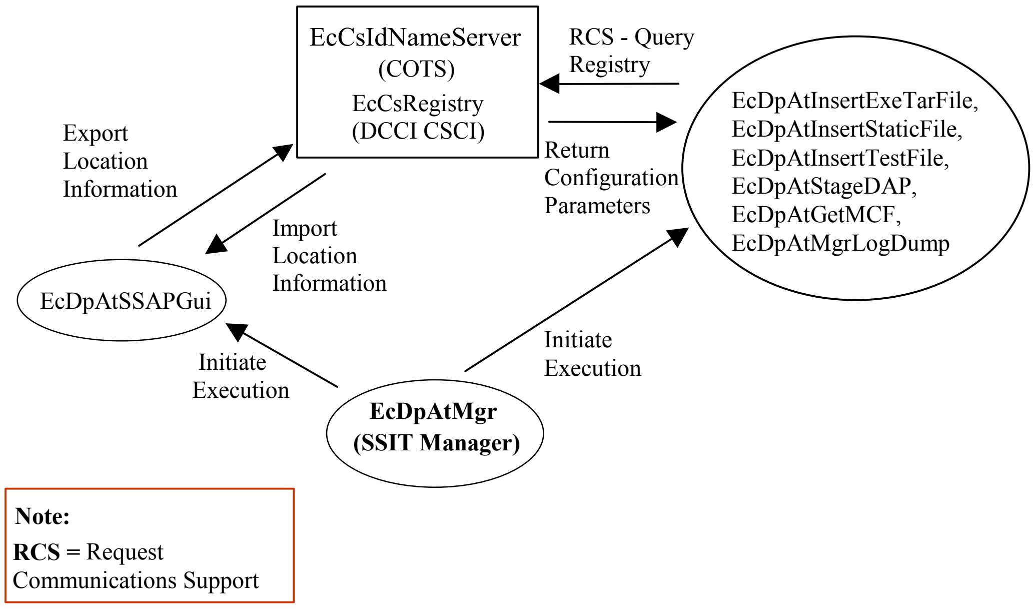
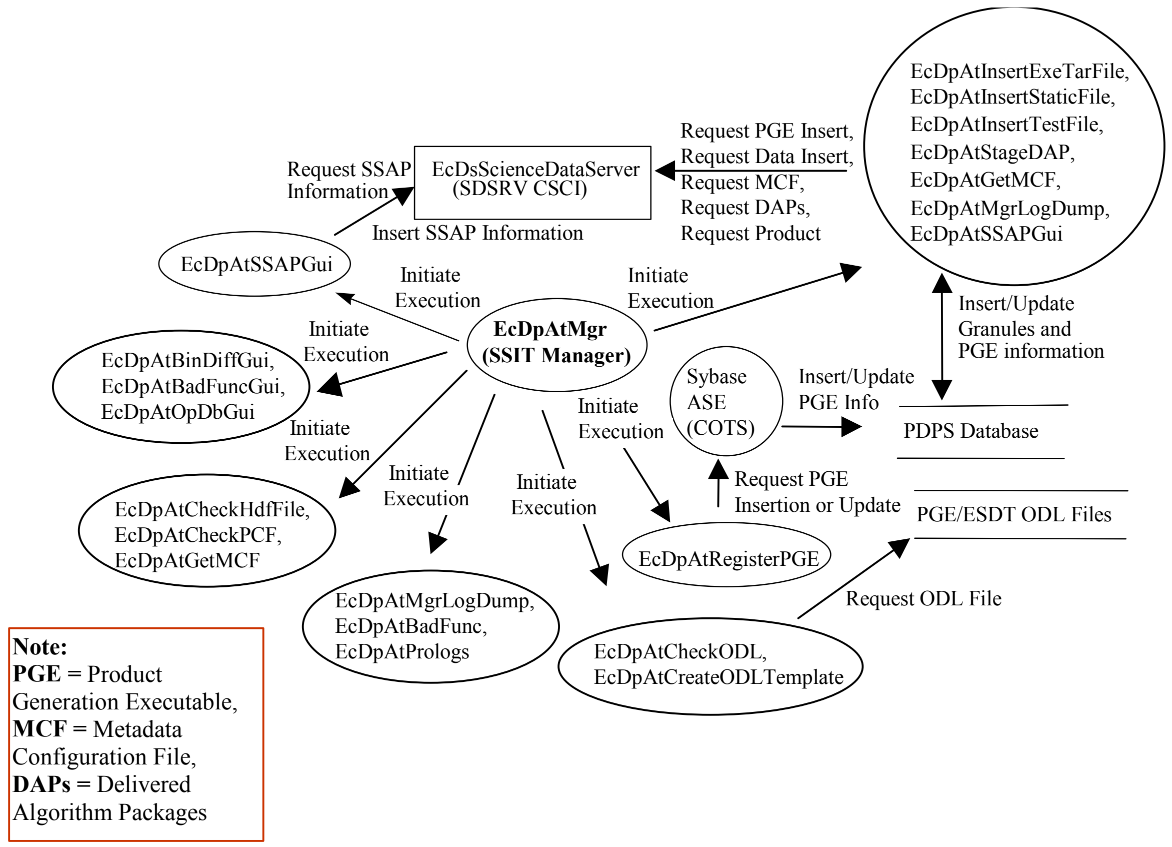

EcCsIdNameServer (COTS)
EcCsRegistry (DCCI CSCI)
RCS - Query Registry
Return Configuration Parameters
EcDpAtInsertExeTarFile, EcDpAtInsertStaticFile, EcDpAtInsertTestFile, EcDpAtStageDAP, EcDpAtGetMCF, EcDpAtMgrLogDump
Export Location Information
Import Location Information
EcDpAtSSAPGui
Initiate Execution
Initiate Execution
EcDpAtMgr (SSIT Manager)
Note:
RCS = Request Communications Support

Figure 4.7-5. AITTL CSCI Architecture Diagram



EcDpAtInsertExeTarFile, EcDpAtInsertStaticFile, EcDpAtInsertTestFile, EcDpAtStageDAP, EcDpAtGetMCF, EcDpAtMgrLogDump, EcDpAtSSAPGui
Request PGE Insert, Request Data Insert, Request MCF, Request DAPs, Request Product
Request SSAP Information
EcDsScienceDataServer (SDSRV CSCI)
Insert SSAP Information
EcDpAtSSAPGui
Initiate Execution
EcDpAtMgr (SSIT Manager)
Insert/Update Granules and PGE information
EcDpAtBinDiffGui, EcDpAtBadFuncGui, EcDpAtOpDbGui
Sybase ASE (COTS)
Insert/Update PGE Info
PDPS Database
EcDpAtCheckHdfFile, EcDpAtCheckPCF, EcDpAtGetMCF
Request PGE Insertion or Update
PGE/ESDT ODL Files
EcDpAtRegisterPGE
EcDpAtMgrLogDump, EcDpAtBadFunc, EcDpAtPrologs
Request ODL File
EcDpAtCheckODL, EcDpAtCreateODLTemplate
Note: PGE = Product Generation Executable, MCF = Metadata Configuration File, DAPs = Delivered Algorithm Packages

Figure 4.7-5. AITTL CSCI Architecture Diagram (cont.)

4.7.2.4 Algorithm and Test Tools Process Description

Table 4.7-8 provides descriptions of the processes shown in the AITTL CSCI architecture diagram.

Table 4.7-8. AITTL Processes (1 of 2)

Process	Type	COTS / Developed	Functionality
EcDpAtSSAPGui	GUI	Developed	This GUI allows the M&O staff to create, update and delete SSAPs and to acquire information about an SSAP for modification or testing (such as test plans).
EcDpAtMgr	GUI	Developed	This application provides menus to launch other SSIT applications and provides a checklist to users for marking each SSIT function as completed.

Table 4.7-8. AITTL Processes (2 of 2)

Process	Type	COTS / Developed	Functionality
EcDpAtRegisterPGE, EcDpAtCheckODL, EcDpAtCreateODLTemplate, EcDpAtOpDbGui	GUI and cmd line interface (I/F)	Developed	This application group allows a PGE to be defined in the PDPS database. ODL is read and checked by the tools and translated into the fields defining a PGE in the PDPS database. If the ODL files are valid, each row already existing in the PDPS database is updated and non-existent rows are inserted. The SSIT personnel input performance information via a GUI.
EcDpAtBinDiffGui, hdiff	GUI and cmd line I/F	Developed and COTS	This application group supports data file viewing and comparisons. The group includes EOSView, the COTS language IDL, and tools to compare binary and HDF files. The shell programs EcDpAtCheckHdfFile and EcDpAtMgrXdiff are used to assist with the viewing and comparisons.
EcDpAtBadFunc, EcDpAtBadFuncGui, EcDpAtPrologs, EcDpAtCheckPCF	GUI	Developed and COTS	This application group checks the source code for PGEs and PGE PCFs for errors or prohibited functions. The Sparc Works COTS product is included for editing and debugging functions and a checker is provided for use during testing. Also provided is a checker to monitor the software for any prohibited calls.
EcDpAtInsertExeTarFile, EcDpAtInsertStaticFile, EcDpAtInsertTestFile, EcDpAtStageDAP, EcDpAtGetMCF	cmd line I/F	Developed	This application group provides mechanisms to insert and acquire data items from the EcDsScienceDataServer in the SDSRV CSCI. Static Files and PGE executables are inserted into the SDSRV archives by these tools, and the respective PDPS database tables are updated with the results. The Delivery Archive Package (DAP) and MCFs are acquired via these tools for command line testing the PGE.
EcDpAtMgrLogDump	Cmd line I/F	Developed	Command line interface to dump the SSIT checks list database to a file that can be sent to the printer.
Sybase ASE	Server	COTS	The Sybase ASE is the interface between AITTL processes and the PDPS database for PGE insertion and update of PGE information in the PDPS database to support Data Processing activities.

4.7.2.5 Algorithm and Test Tools Process Interface Descriptions

Table 4.7-9 provides descriptions of the interface events shown in the AITTL CSCI architecture diagrams.

Table 4.7-9. AITTL Process Interface Events (1 of 7)

Event	Event Frequency	Interface	Initiated By	Event Description
Request Management Services	At system startup or shutdown and for restarts	*Process:* EcDpAtMgr	DAAC unique startup scripts	**System startup and shutdown -** Please refer to the release-related, current version of the Mission Operations Procedures for the ECS Project document (611) and the current ECS Project Training Material document (625), identified in Section 2.2.1 of this document.
Request Communications Support	One service per request	***Process:*** EcCsIdNameServer ***Libraries:*** EcPf, Middleware, FoNs, FoIp, oodce *Classes:* EcPfManagedServer, EcPfClient, CCSMdwNameServer, FoNsNameServerProxy, CCSMdwRwNetProxy ***Library (Common):*** EcUr *Class:* EcUrServerUR ***Library:*** event *Class:* EcLgErrorMsg ***Process:*** EcCsRegistry *Library:* EcCsRegistry *Class:* EcRgRegistryServer_C	*Processes:* EcDpAtSSAPGui, EcDpAtInsertExeTarFile, EcDpAtInsertStaticFile, EcDpAtInsertTestFile, EcDpAtStageDAP, EcDpAtGetMCF *Library:* DpAtDsrv	The **DCCI CSCI** provides a library of services available to each SDPS and CSMS CSCI. The CSCI services required to perform specific assignments are requested from the DCCI CSCI. These services include: • CCS Middleware Support • Database Connection Services • Network & Distributed File Services • Bulk Data Transfer Services • Name/Address Services • Password Services • Server Request Framework (SRF) • Universal Reference (UR) • Error/Event Logging • Fault Handling Services • Mode Information • (Query Registry) Retrieving the requested configuration attribute-value pairs from the Configuration Registry

Table 4.7-9. AITTL Process Interface Events (2 of 7)

Event	**Event Frequency**	**Interface**	**Initiated By**	**Event Description**
Return Configuration Parameters	One set per request	*Processes:* EcDpAtInsertExeTarFile, EcDpAtInsertStaticFile, EcDpAtInsertTestFile, EcDpAtStageDAP, EcDpAtGetMCF, EcDpAtMgrLogDump	*Process:* EcCsRegistry *Library:* EcCsRegistry *Class:* EcRgRegistryServer_C	The **EcCsRegistry** returns the attribute-value pairs (configuration parameters) to the EcDmDictServer upon request.
Initiate Execution	One per tool initialization	UNIX system calls	*Process:* EcDpAtMgr *Library:* DpAtDsrv *Class:* DpAtMgrLogGuiMainWindow	The EcDpAtMgr initiates the tools (**EcDpAtBinDiffGui, EcDpAtBadFuncGui, EcDpAtOpDbGui, EcDpAtCheckHdfFile, EcDpAtCheckPCF, EcDpAtGetMCF, EcDpAtMgrLogDump, EcDpAtBadFunc, EcDpAtPrologs, EcDpAtCheckODL, EcDpAtCreateODLTemplate, EcDpAtRegisterPGE, EcDpAtInsertExeTarFile, EcDpAtInsertStaticFile, EcDpAtInsertTestFile, and EcDpAtStageDAP**) and the GUI interface (**EcDpAtSSAPGui**) from a menu.

Table 4.7-9. AITTL Process Interface Events (3 of 7)

Event	Event Frequency	Interface	Initiated By	Event Description
Export Location Information	Once at system startup and after each failure recovery	*Process:* EcCsIdNameServer *Library:* EcPf, Middleware, FoNs, FoIp, oodce *Classes:* EcPfManagedServer, EcPfClient, CCSMdwNameServer, FoNsNameServerProxy, CCSMdwRwNetProxy	*Process:* EcDpAtSSAPGui	The EcDpAtSSAPGui places physical and logical location information in the **EcCsIdNameServer**.
Import Location Information	As required for processing	*Process:* EcDpAtSSAPGui	*Process:* EcCsIdNameServer *Library:* EcPf, Middleware, FoNs, FoIp, oodce *Classes:* EcPfManagedServer, EcPfClient, CCSMdwNameServer, FoNsNameServerProxy, CCSMdwRwNetProxy	The EcDpAtSSAPGui requests server location information from the **EcCsIdNameServer**.

Table 4.7-9. AITTL Process Interface Events (4 of 7)

Event	Event Frequency	Interface	Initiated By	Event Description
Request PGE Insert	One per insert request	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClRequest, DsClCommand	*Processes:* EcDpAtInsertExeTarFile, EcDpAtInsertStaticFile, EcDpAtInsertTestFile *Library:* PlCore2 *Classes:* DpAtDsrv, PlResourceRequirement	The EcDpAtInsertExeTarFile, EcDpAtInsertStaticFile, and EcDpAtInsertTestFile send PGE insert requests to the **EcDsScienceDataServer** for data that defines a PGE and allows it to be scheduled and executed.
Request Data Insert	One per data put into the archive	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClRequest, DsClCommand	*Processes:* EcDpAtInsertTestFile, EcDpAtInsertExeTarFile, EcDpAtInsertStaticFile, EcDpAtSSAPGui *Library:* DpAtDsrv *Class:* DpAtDsrv	The EcDpAtInsertStaticFile, EcDpAtInsertTestFile, EcDpAtInsertExeTarFile, and EcDpAtSSAPGui processes send requests to the **EcDsScienceDataServer** to put various types of data into the archive, from SSAP information to Static files and PGE executables.
Request MCF	One per MCF request	*Process:* EcDsScienceDataServer *Library:* DsCl *Class:* DsClDescriptor	*Process:* EcDpAtGetMCF *Library:* DpAtDsrv *Class:* DpAtDsrv	The EcDpAtGetMCF process sends a request for a MCF template to the **EcDsScienceDataServer**. In response, the MCF template is returned and populated.
Request DAPs	One per DAPs request	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClESDTReferenceCollector, DsClRequest, DsClCommand	*Process:* EcDpAtStageDAP *Library:* DpAtDsrv *Class:* DpAtDsrv	The EcDpAtStageDAP requests DAPs from the SDSRV Archives (at the **EcDsScienceDataServer**) based on the UR. In response, the DAPs are returned and stored on the local AITTL disk.

Table 4.7-9. AITTL Process Interface Events (5 of 7)

Event	Event Frequency	Interface	Initiated By	Event Description
Request Product	One per user request	Process: EcDsScienceDataServer *Library:* DsDdSSh *Classes:* DsDdScheduler, DsDdRequestMgrReal	*Process:* EcDpAtStageDAP *Library:* DpAtDsrv *Class:* DpAtDsrv	The EcDpAtStageDAP sends requests to the **EcDsScienceDataServer** for particular data granules to be pushed, via the FTP service, onto the DPS science processor as input for data processing or for SSIT work.
Return MCF Template	One per set of external data received by ECS	*Process:* EcDpAtGetMCF *Library:* DpPrDssIF *Class:* DpPrDSSInterface	*Process:* EcDsScienceDataServer *Library:* DsCl *Class:* DsClDescriptor	The **EcDsScienceDataServer** provides the MCF template as part of the GetMCF service call to the EcDpAtGetMCF process.
Return DAPs	One DAP per request	*Process:* EcDpAtStageDAP *Library:* DpAtDsrv *Class:* DpAtDsrv	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClESDTReferenceCol lector, DsClRequest, DsClCommand	The **EcDsScienceDataServer** returns DAPs from the SDPS archives to the EcDpAtStageDAP, which stores the DAPs on the local AITTL disk.
Return SSAP Information	One per SSAP information request	*Process:* EcDpAtSSAPGui *Libraries:* DpAtSSAP, DpAtDsrv *Classes:* DpAtSSAPManager, DpAtDsrv	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClESDTReferenceCol lector, DsClRequest, DsClCommand	The **EcDsScienceDataServer** returns information about SSAPs, including names of existing SSAPs and the components associated with a specific SSAP to the EcDpAtSSAPGui.

Table 4.7-9. AITTL Process Interface Events (6 of 7)

Event	Event Frequency	Interface	Initiated By	Event Description
Return Product	One per product order request	*Processes:* EcDpAtInsertExeTarFile, EcDpAtInsertStaticFile, EcDpAtInsertTestFile *Library:* DpAtDsrv *Class:* DpAtDsrv	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClRequest, DsClCommand	The **EcDsScienceDataServer** returns data granules to the EcDpAtInsertExeTarFile, EcDpAtInsertStaticFile and EcDpAtInsertTestFile processes as input for data processing or for SSIT work.
Insert/ Update Granules and PGE information	One per insert/update of granule information	PDPS Database	*Processes:* EcDpAtInsertStaticFile, EcDpAtInsertExeTarFile *Library:* PlCore1 *Classes:* DpAtDsrv, PlDataGranule	Insert/update granule information in the **PDPS Database**: • Received from a static granule insert request • About a modified, existing PGE
Insert/Update PGE Info	One PGE per request	PDPS Database	Sybase ASE (COTS)	The **Sybase ASE** inserts or updates PGE information in the PDPS database.
Request PGE Insertion or Update	One per insert/update request	Sybase ASE (COTS)	*Process:* EcDpAtRegisterPGE *Library:* PlCore2 *Class:* PlResourceRequirement	The EcDpAtRegisterPGE process sends insert or update requests to the **Sybase ASE** to add or modify PGE information in the PDPS database to perform data processing tasks.
Request ODL File	One per ODL file request	PGE/ESDT ODL Files	*Processes:* EcDpAtCheckODL, EcDpAtRegisterPGE *Library:* DpAtMetadata *Classes:* DpAtDatabase, DpAtScienceMd	In response to a request for an **ODL File**, the EcDpAtCheckODL and EcDpAtRegisterPGE processes receive data in "parameter = value" format about a **PGE**, its inputs and outputs, and scheduling information.

Table 4.7-9. AITTL Process Interface Events (7 of 7)

Event	Event Frequency	Interface	Initiated By	Event Description
Initiate Execution	One per tool initialization	*Processes:* EcDpAtInsertExeTarFile, EcDpAtInsertStaticFile, EcDpAtInsertTestFile, EcDpAtStageDAP, EcDpAtGetMCF, EcDpAtMgrLogDump, EcDpAtRegisterPGE, EcDpAtCheckODL, EcDpAtCreateODLTemplate, EcDpAtBadFunc, EcDpAtPrologs, EcDpAtCheckHdfFile, EcDpAtCheckPCF, EcDpAtBinDiffGui, EcDpAtBadFuncGui, EcDpAtOpDbGui, EcDpAtSSAPGui	*Process:* EcDpAtMgr *Library:* DpAtDsrv *Class:* DpAtMgrLogGuiMainWindow	The EcDpAtMgr initiates the tools (**EcDpAtInsertExeTarFile, EcDpAtInsertStaticFile, EcDpAtInsertTestFile, EcDpAtStageDAP, EcDpAtGetMCF, EcDpAtMgrLogDump, EcDpAtRegisterPGE, EcDpAtCheckODL, EcDpAtCreateODLTemplate, EcDpAtBadFunc, EcDpAtPrologs, EcDpAtCheckHdfFile, and EcDpAtCheckPCF**) and the GUI interfaces (**EcDpAtBinDiffGui, EcDpAtBadFuncGui, EcDpAtOpDbGui, and EcDpAtSSAPGui**) from a menu using UNIX system calls.
Request SSAP Information	One per SSAP information request	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClRequest, DsClCommand, DsClESDTReferenceCollector	*Process:* EcDpAtSSAPGui *Libraries:* DpAtSSAP, DpAtDsrv *Classes:* DpAtSSAPManager, DpAtDsrv	The EcDpAtSSAPGui sends requests to the **EcDsScienceDataServer** for information about SSAPs, including names of existing SSAPs and the components associated with a specific SSAP.
Insert SSAP Information	One per SSAP	*Process:* EcDsScienceDataServer *Library:* DsCl *Classes:* DsClRequest, DsClCommand	*Process:* EcDpAtSSAPGui *Library:* DpAtDsrv *Class:* DpAtSSAPManager, DpAtDsrv	The EcDpAtSSAPGui sends requests to the **EcDsScienceDataServer** to insert new SSAP information or update any existing SSAP information.

4.7.2.6 Algorithm and Test Tools Data Stores

Table 4.7-10 provides descriptions of the data stores shown in the AITTL CSCI architecture diagram.

Table 4.7-10. AITTL Data Stores

Data Store	Type	Functionality
PGE/ESDT ODL files	Files	These files are written in *parameter* = *value* formats to define the inputs and outputs of a PGE and any relevant scheduling information (including Production Rules), and are created by the Instrument Teams and the SSIT personnel.
PDPS database	Database	The PDPS database is replicated at each site for fault handling and recording purposes. The PDPS database holds all the persistent data including: • Resource information entered with the Resource Planning utilities • PGE and data type information entered at SSIT • Production Request, Data Processing Request and Data Granule information entered using the Production Request Editor • Plan information entered using the Production Planning Workbench

4.7.3 Data Processing Hardware Components

4.7.3.1 Science Processor Hardware CI (SPRHW) Description

Science Processor hardware (SPRHW) consists of the Science Processor Hardware and the Queuing Server Hardware.

The Science Processor Hardware features Redundant Arrays of Inexpensive Disks (RAID) devices set at RAID Level 3. The Queuing Server Hardware features attached disk packs for additional storage. X-terminals are also provided as part of the SPRHW for additional user access to ECS.

Science Processor

The Science Processors are connected to the Production network via switched Ethernet. The Science Processor is based on a 64-bit SGI machine. Each Science Processor consists of 12 or 16 processors (See 920-TDx-001 series of base-line documents to see how the configuration is determined). Each Science Processor has one to six GB of memory with eight-way interleaving to improve the input/output (I/O) performance between the processors and memory (see 920-TDx-001 series of base-line documents). The SGI architecture is configured with I/O subsystems attached to the back plane and referred to as PowerChannel2 or IO4 cards. Each IO4 provides serial and parallel connections, two fast-wide differential SCSI-2 channels, and space for two High Input Output (HIO) controller cards. HIO controller card includes a high-speed Gigabit Ethernet card, an Ethernet card, and a card to support three SCSI-2 channels.

The number of IO cards specified for each Science Processor is determined by allocating HIO slots to the Ethernet and high-speed Gigabit Ethernet interfaces, and counting the number of SCSI-2 interfaces required. The number of internal and external SCSI-2 devices supported by the system determines the required number of SCSI-2 interfaces. The first SCSI-2 channel is delegated to internal devices, i.e., CD-ROMs, floppy disk drives, and tape drives. Internal disks ranging in aggregate size from eight GB to 12 GB are allocated to the second SCSI-2 channel. External disk arrays are allocated to subsequent SCSI-2 channels; the number of channels is based on the required throughput of the external file systems (see 920-TDx-001 and 922-TDx-015 series of base-line documents).

The internal disks of the Science Processor are only used to provide swap space for the operating system and to provide file system space for the operating system and applications (see 920-TDx-001 and 922-TDx-015 series of base-line documents).

Data transfer requirements between the Science Processor and the DSS are met with a switched high-speed Gigabit Ethernet network implemented via a central high-speed Gigabit Ethernet switch with switched 800 Mbps interface ports. The Science Processor connects directly to the DSS hosts (see 920-TDx-001, 921-TDx-002, 921-TDx-003, and 921-TDx-004 series of base-line documents).

Because the Science Processor does not provide long term, secure storage, data backup and recovery are not provided. The Science Processor storage is used to hold ancillary data files and data granules for short periods of time. If a file system failure occurs within a Science Processor, the algorithms, ancillary data files, and data granules are recoverable from the Data Server.

Failure recovery for the high-speed Gigabit Ethernet switch used is supplied by stocking spare Line Replaceable Units of the switch power supplies, interface cards, fan. If an individual interface card fails, a host is re-configured to a hot spare interface card by moving two cables and re-configuring the switch. If the control module fails, it is replaced with a spare module and the switch is re-configured. In the event of a failure of the entire switch, the switch is either replaced or repaired.

Queuing Server

The Queuing Server is connected to the Production network via switched Ethernet. The Queuing Server with a Sybase ASE database directs AutoSys to load and execute the daily production schedule of a DAAC.

The Queuing Server is based on the SUN Server or the SUN workstation depending on the DAAC site capacity requirements. With a load requirement on the AutoSys and Sybase database for a 24-hour production run of 187,200 jobs, the Queuing Server uses four Ultra-SPARC processors. DAAC sites with smaller production runs are equipped with a Queuing Server based on a dual-processor, SPARC-based workstation (see 920-TDx-001 series of baseline documents).

Each Queuing Server is equipped with a minimum of 384 MB of memory to meet the AutoSys and Sybase ASE database processing requirements (see 920-TDx-001 series of baseline documents).

The internal disks on a Queuing Server are only used to provide swap space for the operating system and to provide file system space for the operating system and applications (see 920-TDx-001 and 922-TDx-014 series of baseline documents).

Additional storage needed to support the Sybase ASE database and to back-up the database from the Planning and Data Processing Subsystems (PDPS) Database Management System Server is via a SCSI-2 interface. To support failure recovery of the Sybase ASE databases, two times the normal operating storage is available (see 920-TDx-001 and 922-TDx-014 series of baseline documents).

The AutoSys database located on the Queuing Server is replicated by the MSS Backup Server (See Section 4.9.4.1: MHCI Description) to a physical location on the PDPS Database Management System Server. When a disk or database failure occurs on the primary database, AutoSys continues to operate using the backup database on the PDPS Database Management System Server.

4.7.3.2 Algorithm Quality Assurance Hardware CI Description

The Algorithm Quality Assurance workstations are connected to the Production network via switched Ethernet. Algorithm Quality Assurance Hardware (AQAHW) used to validate the quality of ECS products includes non-science QA, in-line QA, and SCF-based QA. Non-science QA is specified by the DAAC Operations staff and includes data integrity checks on the data products and the metadata. In-line QA is a form of science QA validating product content using science algorithms. The ECS provides support for SCF-based QA by providing archive and communications capacity for the SCFs to sample and validate the contents of the products.

The AQAHW is an AQA workstation and a Disk/RAID Driver.

AQA Workstation

The AQA workstation provides a software execution environment equivalent to the AI&T software execution environment in order to facilitate the use of the AQA workstation for AI&T when necessary. Also, the AQA supports complex data viewing techniques.

The AQA workstation is a 64-bit SGI machine. For information on the processors used, see the 920-TDx-001 series of base-line documents. The AQA workstation is equipped with a minimum of 128 MB of memory. The AQA workstation is equipped with four EISA slots. These EISA slots have a transfer rate of 33 MB per second. Additionally, the AQA workstation is equipped with two fast SCSI-2 connections.

The internal disk provides swap space for the operating system and file system space for the operating system and applications (see 920-TDx-001 series of base-line documents). There are no external storage arrays.

AQA Disk/RAID Driver

The AQA Disk/RAID Driver supports the AQA Workstation by providing storage for the QA and SCF-based QA activities.

The AQA Disk/RAID Driver is a 64-bit SGI machine. For information on the processors, see the 920-TDx-001 series of base-line documents. The AQA Workstation is equipped with a minimum of 128 MB of memory.

The internal disk provides swap space for the operating system and file system space for the operating system and applications (See 920-TDx-001 and 922-TDx-003 series of base-line documents).

SGI storage units referred to as "Vaults" are attached to the AQA Disk/RAID Driver, via a SCSI-2 interface, to provide the additional storage space (See 920-TDx-001 and 922-TDx-003 series of base-line documents) to support QA.

4.7.3.3 Algorithm Integration and Test Hardware CI Description

The Algorithm Integration and Test Hardware (AITHW) Configuration Item is the hardware to support the system level software validation, integration, and test and the integration and test of science software at a DAAC.

AITHW contains an AIT workstation and an AIT/Sybase ASE with a laser printer and X-terminals to provide additional user access.

AIT Workstation

The AIT workstation connects to the Production network via switched Ethernet. The AIT workstation is a 64-bit SUN workstation class machine with 128 MB of memory (See 920-TDx-001 series of base-line documents). The AIT workstation is for building and testing software in the AIT environment.

The AIT/Sybase ASE internal disk provides swap space for the operating system and file system space for the operating system and applications (See 920-TDx-001 series of base-line documents). An external disk pack is attached via the SCSI port to provide additional storage.

The function of this component is not critical to data processing. In the event of a component failure, the faulty component is either replaced or repaired by a certified SUN technician.

AIT/Sybase ASE

The AIT workstation/Sybase ASE connects to the Production network via switched Ethernet. The AIT/Sybase ASE is the hardware tools and database support for AIT.

The AIT/Sybase ASE is a 64-bit SUN workstation class machine. For information on the processor utilized in this workstation see the 920-TDx-001 series of base-line documents. The AIT/Sybase ASE is equipped with a minimum of 256 MB of memory (See 920-TDx-001 series of base-line documents).

The AIT internal disk provides swap space for the operating system and file system space for the operating system and applications (See 920-TDx-001 series of base-line documents) with external multi-packs attached via the SCSI port to provide additional storage.